Université de Montréal

Partition adaptative de l'espace dans un algorithme MCMC avec adaptation régionale

par

Nicolas Grenon-Godbout

Département de mathématiques et de statistique
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en statistique

15 juin 2018

SOMMAIRE

La simulation de variables aléatoires provenant de lois multimodales par des méthodes MCMC présente des défis particuliers. Les algorithmes adaptatifs utilisés pour faire face à ces distributions cherchent à faire le bon compromis entre efficacité asymptotique et vitesse d'exécution. Nous proposons dans ce mémoire une amélioration d'un algorithme MCMC avec adaptation régionale (RAPT) de Craiu et al. (2009), qui consiste à ajouter à ce dernier une partition adaptative régularisée de l'espace échantillonnal. Précisément, la partition est déterminée en construisant un hyperplan perpendiculaire à la droite joignant les moyennes échantillonnales cumulées dans chaque région, et passant par le point équidistant des deux moyennes selon la distance de Mahalanobis. Le but de cet ajout est de conférer plus de robustesse par rapport à la partition initiale des régions, tout en ajoutant un coût computationnel minimal à la procédure.

Nous détaillons le fonctionnement de l'algorithme et de ses variantes, et situons celui-ci dans le cadre général des méthodes MCMC adaptatives, incluant une revue des principaux algorithmes avec adaptation régionale compétiteurs. L'ergodicité du processus généré par ce nouvel algorithme est démontrée à l'aide des conditions suffisantes d'ergodicité de Roberts et Rosenthal (2007).

Nous présentons des illustrations graphiques du processus d'adaptation de la partition. Puis, nous comparons les performances de notre algorithme à celles des algorithmes RAPT et RAPTOR à l'aide de nombreuses simulations inspirées de la littérature et dégageons les forces et faiblesses de ces trois options. En général, notre algorithme a offert des résultats comparables à ceux de ses compétiteurs et s'est avéré dans certains cas le meilleur choix, surtout lorsque l'on tient compte de l'effort computationnel requis.

Mots-clés : MCMC, algorithmes adaptatifs, adaptation régionale, partition, ergodicité, distributions bimodales

SUMMARY

Sampling from multimodal distributions using MCMC methods is an ongoing challenge. Most adaptive algorithms aim to find a good compromise between asymptotic efficiency and actual computation time. In this masters thesis, we propose an improvement to the regional adaptive(RAPT) MCMC algorithm of Craiu et al. (2009), that consists in adding a constrained adaptive partitionning of the sample space. Precisely, the adaptive partition is defined as the hyperplane that is orthogonal to the line joining the cumulative sample means in each region, and which passes through the equidistant point between those two means, according to the Mahalanobis distance. We thus obtain an algorithm that is robust to the choice of the initial partition, at a mimimal computational cost.

The workings of the algorithm and its variants are detailed and set in the general framework of adaptive MCMC. A review of some widely used regional adaptation algorithms is also included. The ergodicity of the generated sampler is proved using the sufficient ergodicity conditions of Roberts et Rosenthal (2007).

Graphical illustrations of the adaptive partitionning are presented. Our approach is then compared to the RAPT and RAPTOR algorithms through many examples inspired by MCMC litterature. Our new algorithm yields satisfactory results and is in some cases the best choice among those considered, especially when taking into account the computational effort.

Keywords : MCMC, adaptive algorithms, regional adaptation, partitionning, ergodicity, bimodal distributions

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET DES ABRÉVIATIONS

AQV	Variation quadratique moyenne, de l'anglais *Average Quadratic Variation*
i.i.d.	Indépendantes et identiquement distribuées (à propos de variables aléatoires)
MCMC	Méthodes de Monte-Carlo par chaîne de Markov, de l'anglais *Markov Chain Monte Carlo*
EQM	Erreur quadratique moyenne
MH	Algorithme de Metropolis-Hastings ([19], [22])
AM	*Adaptive Metropolis*, algorithme MCMC adaptatif de [16]
RAPT	*Regional adAPTive algorithm*, algorithme MCMC adaptatif avec partition régionale de [6]
RAPTOR	*Regional adAPTive algorithm with Online Recursion*, algorithme MCMC adaptatif avec partition régionale adaptative de [3]
OPRA	*Online Partitioning of the Regional Adaptive algorithm*, notre algorithme MCMC adaptatif avec partition régionale adaptative

REMERCIEMENTS

Je souhaite tout d'abord remercier ma superviseure, Mylène Bédard, pour son implication tout au long de ce projet. Elle a toujours su allier fluidité et rigueur, en me laissant la liberté de choisir ma propre voie tout en assurant avec sérieux le suivi de mon travail. Elle est allée bien au-delà des exigences de son mandat en m'apportant réponses éclairantes, conseils judicieux et réflexions intéressantes lors de nos longues discussions, dont le sujet dépassait bien souvent le cadre de ce mémoire et de la statistique en général.

Merci d'avoir été une guide exemplaire, et surtout une personne splendide.

Je remercie également le personnel enseignant et administratif du département pour leur travail efficace et le support qu'ils apportent aux étudiants. Je suis particulièrement reconnaissant envers les professeurs Christian Léger et Maciej Augustyniak, non seulement pour la qualité de leur travail en tant que membres du jury, mais aussi pour leur enseignement à la fois stimulant et exigeant.

Mes remerciements vont aussi au Conseil de recherches en sciences naturelles et en génie du Canada et aux Fonds de recherche du Québec - nature et technologies, pour l'apport financier qui a permis de mener à bien ce projet.

Je ne peux m'empêcher de souligner la contribution inestimable des nombreux collègues, dont plusieurs sont devenus de précieux amis, que j'ai eu la chance de côtoyer tout au long de mon parcours universitaire. Chacun et chacune à leur façon, ils transforment à chaque jour ce bâtiment de briques à l'intégrité variable en un endroit magique où il fait bon apprendre, rire et partager. Il serait impossible de tous les nommer ici, mais je m'en voudrais de ne pas formuler quelques mentions spéciales.

Rébecca, pour avoir été une amie sincère, une coéquipière hors-pair et bien plus encore ; Stéphanie, pour ton énergie si unique et ta capacité d'être toujours là au bon moment ; François-Simon, pour ta générosité et ta force tranquille ; Gabrielle, pour toutes les choses que tu m'as apprises ; Justin, pour toutes ces soirées mémorables au CEPSUM ; Alexis, pour ta belle érudition, ainsi que ta lecture plus qu'attentive de mon manuscrit ; David, pour

ton authenticité, tant capillaire que générale ; Catherine, Julie, Vanessa, Victoire, Mikaël, Michaël, Fabrice, Mireille, Jérémie, Vincent, Antoine, Louis-Marc ...Ce fut un plaisir et un privilège d'avoir vingt ans une seconde fois en votre compagnie.

J'envoie mes salutations aussi à tous les membres de la troupe du CoMUM et du Phoenix, pour leur accueil, leur amitié et leur folie.

Danita, merci pour ta présence, ta patience, ton rire, tes encouragements. Cette aventure n'aurait pas été la même sans toi.

Je termine avec une pensée toute particulière pour mes parents et ma famille, pour leur support ininterrompu et leur amour inconditionnel.

INTRODUCTION

Les méthodes de Monte-Carlo par chaînes de Markov (MCMC) sont une classe d'algorithmes permettant de simuler des variables aléatoires provenant de distributions complexes, d'abord utilisées en physique statistique (voir [22]) puis développées subséquemment de façon plus générale par [19]. Elles sont largement utilisées en applications de la statistique bayésienne, dans des domaines aussi variés que l'analyse de séquences moléculaires en biologie évolutive (voir [8]) et la paramétrisation des modèles climatiques (voir [31]).

Leurs aspects théorique et pratique sont ainsi l'objet de nombreuses études. Notamment, plusieurs recherches ont tenté de déterminer une calibration optimale de ces algorithmes en fonction d'une distribution cible donnée, afin de maximiser leur efficacité. Cette calibration a été identifiée sous certaines conditions théoriques, et est suffisamment robuste à une violation de ces dernières. Toutefois, elle dépend d'une bonne connaissance de la forme et de l'échelle de la distribution que l'on cherche à simuler. Ceci occasionne une situation paradoxale, étant donné que l'on cherche à simuler efficacement cette distribution justement parce qu'il est difficile d'avoir des informations sur celle-ci autrement.

Cette situation a mené au cours des vingt dernières années au développement d'algorithmes MCMC dits adaptatifs. Basés en grande partie sur le travail de [16], ces derniers ont pour particularité de mettre automatiquement à jour leur calibration à mesure que l'algorithme progresse et fournit de nouvelles informations sur la distribution à simuler. Autrement dit, d'une certaine façon, ces algorithmes apprennent la distribution cible en cours de procédure et utilisent cette connaissance pour améliorer leur paramétrisation au fil du temps.

Plus récemment encore, de nouveaux algorithmes, avec adaptation dite régionale, ont été introduits dans le but de simuler efficacement des distributions que l'on pourrait qualifier de plus exigeantes, pensons par exemple à des lois bimodales ou asymétriques. Ces algorithmes de dernière génération ont comme particularité d'utiliser différentes paramétrisations en fonction de l'emplacement dans l'espace de l'état actuel de la simulation, d'où leur nom. Deux algorithmes populaires de ce genre sont le RAPT et le RAPTOR, développés dans [6] et [3] respectivement. Le premier, relativement efficace, est toutefois sensible à un mauvais choix des paramètres initiaux, en particulier la frontière fixe délimitant les différentes

régions de l'espace. Le second, conçu entre autres choses pour pallier à ce problème, permet non seulement l'adaptation de ses paramètres dans chaque région, mais aussi l'adaptation de l'emplacement des régions elles-mêmes. C'est donc un algorithme très flexible, mais qui a comme défaut d'être plus demandant au niveau informatique, donc plus lent.

Nous proposons dans ce mémoire un troisième algorithme, OPRA, qui se veut un compromis entre ces deux options. L'idée est d'ajouter à l'algorithme RAPT une composante permettant l'adaptation de l'emplacement des régions. Toutefois, afin de préserver autant que possible la vitesse de l'adaptation, la forme de la frontière entre les régions est limitée à des hyperplans. Nous espérons ainsi obtenir un processus plus efficace que le RAPT mais de vitesse similaire, et qui se comparerait avantageusement au RAPTOR en terme de temps réel de calcul.

Ce mémoire résume donc le travail effectué pour valider la pertinence de cet algorithme, tant au niveau théorique que pratique. Nous verrons, à l'aide de conditions de convergence sur les MCMC adaptatives développées dans [28], que le processus généré par OPRA converge effectivement vers la distribution d'intérêt. Dans un second temps, nous étudierons le comportement de notre algorithme sur de nombreux exemples simulés numériquement. Les résultats obtenus permettent d'affirmer qu'OPRA a un comportement stable et offre des performances compétitives dans la plupart des circonstances, et mettent aussi en évidence certaines faiblesses insoupçonnées de l'algorithme RAPTOR.

Le chapitre 1 est dédié au rappel de certaines notions liées aux deux disciplines ayant donné leur nom aux MCMC, soit les chaînes de Markov et les méthodes de calcul basé sur des processus aléatoires, dites de Monte-Carlo. Nous y effectuons un survol du cadre de travail de l'analyse bayésienne, qui permettra de motiver la pertinence des MCMC dans le développement de ce domaine. Par la suite, le chapitre 2 décrit le fonctionnement de base des MCMC, notamment par la présentation de l'algorithme fondateur de Metropolis-Hastings, point de départ commun de presque tous les algorithmes qui ont suivi. La seconde portion de ce chapitre introduit les méthodes MCMC adaptatives et certains résultats théoriques de convergence liés à celles-ci.

Suite à cette mise en contexte, nous plongeons au coeur du sujet avec le chapitre 3, présentant en détail trois algorithmes MCMC avec adaptation régionale, soient RAPT, RAPTOR et notre propre algorithme, OPRA. Le chapitre 4 est de nature essentiellement théorique et sert à démontrer la convergence du nouvel algorithme proposé sous des conditions suffisamment larges. Par le fait même, de nombreux autres résultats sur les algorithmes adaptatifs y sont démontrés, incluant une généralisation de la preuve de convergence de l'algorithme RAPT.

La dernière portion du mémoire a pour but de comparer les performances empiriques des trois méthodes avec adaptation régionale présentées. Le chapitre 5 définit plusieurs critères de qualité servant à mesurer le degré de convergence et à mettre en rapport différents algorithmes MCMC. Le chapitre 6 contient une analyse des résultats des nombreuses simulations effectuées pour situer notre algorithme par rapport à ses compétiteurs en terme de performance. Finalement, dans le chapitre 7, nous résumons les points importants de cette recherche et formulons certaines recommandations pratiques et pistes d'exploration futures.

Chapitre 1

NOTIONS PRÉLIMINAIRES

Nous verrons dans ce chapitre quelques résultats classiques sur les chaînes de Markov, permettant d'éclaircir le fonctionnement et le comportement des méthodes présentées tout au long du mémoire. Nous introduirons ensuite brièvement les méthodes Monte-Carlo en général et l'analyse bayésienne, dans le but de définir le cadre dans lequel les méthodes MCMC sont habituellement utilisées et afin de justifier la pertinence de ces dernières.

1.1. Chaînes de Markov

Nous résumons ici quelques propriétés des chaînes de Markov qui seront utiles pour la suite. Celles-ci concernent surtout le comportement à long terme des chaînes de Markov et les conditions qui y sont associées.

1.1.1. Définition

On appelle chaîne de Markov à temps discret toute suite de variables aléatoires $\{X_t\}_{t\geq 0}$, provenant d'un certain support S et respectant la propriété markovienne :

$$\mathbb{P}(X_t|X_{t-1},\ldots,X_0) = \mathbb{P}(X_t|X_{t-1}).$$

Autrement dit, la valeur de cette suite à un temps donné, sachant toutes les valeurs précédentes, ne dépend en fait que de la valeur du temps précédent. Nous nous intéresserons ici uniquement aux chaînes dites homogènes, où les probabilités $\mathbb{P}(X_t|X_{t-1})$ sont indépendantes de la valeur de t. Ainsi, sur un espace discret, une chaîne de Markov homogène est complètement définie par la valeur ou la distribution de X_0 et les probabilités de transition en un pas $p_{ij} := \mathbb{P}(X_{t+1} = j|X_t = i), t \geq 0$, habituellement regroupées dans une matrice $P = \{p_{ij}\}$, définie sur l'ensemble des valeurs i,$j \in S$ (l'espace d'états). Finalement, on définit les probabilités de transition en k pas ainsi : $p_{ij}^{(k)} := \mathbb{P}(X_{t+k} = j|X_t = i)$.

1.1.2. Quelques critères de classification

Pour qu'une chaîne ait des propriétés asymptotiques intéressantes, on souhaitera que tous ses états soient récurrents positifs et apériodiques. On dira alors que la chaîne est ergodique. Un état i est dit récurrent positif si l'espérance du temps de retour à l'état i lorsqu'on part de l'état i est finie. Il sera souvent plus facile de vérifier le critère équivalent suivant,

$$p_{ii}^{(n)} \nrightarrow 0, \text{ quand } n \to \infty.$$

Un état est apériodique si sa période vaut 1, la période étant définie comme étant le plus grand commun diviseur de $\{n \in \mathbb{N} : p_{ii}^{(n)} > 0\}$. Pour vérifier ces propriétés, il sera plus simple de passer par la notion de classe d'états, que nous définissons maintenant.

On dit qu'un état j est accessible à partir d'un état i, s'il est possible, en partant de i, d'atteindre l'état j en un nombre fini de transitions. Formellement,

$$i \to j \Leftrightarrow \exists\ n \geq 0, \text{ tel que } p_{ij}^{(n)} > 0.$$

Si l'on a à la fois $i \to j$ et $j \to i$, on dira que i et j communiquent. On montre alors facilement que les états communiquant entre eux forment une classe d'équivalence sur l'ensemble des états de la chaîne. Les états d'une même classe possèdent plusieurs propriétés communes. Entre autres, ils auront la même période et seront tous récurrents positifs si l'un d'entre eux l'est. Ainsi, une chaîne irréductible (possédant une seule classe d'états) et apériodique possédant au moins un état récurrent positif sera ergodique.

1.1.3. Théorème ergodique et distribution stationnaire

La distribution stationnaire, lorsqu'elle existe, est une distribution de probabilités prenant valeur sur S et notée habituellement par le vecteur ligne $\pi := (\pi_i)_{i \in S}$, où $\pi_i = \mathbb{P}(Z = i)$, pour une variable Z de loi π, et respectant la condition $\pi = \pi P$. Autrement dit :

$$\forall j : \pi_j = \sum_i \pi_i p_{ij}$$

Dans le cas d'une chaîne ergodique, la distribution stationnaire existera toujours et sera unique. De plus, on aura alors, pour tous les états i,$j \in S$:

$$p_{ij}^{(n)} \xrightarrow[n \to \infty]{} \pi_j > 0.$$

Il s'agit du théorème ergodique pour les chaînes de Markov homogènes sur un espace discret. Ainsi, une chaîne de Markov ergodique convergera en loi vers sa distribution stationnaire peu importe la valeur initiale X_0.

1.1.4. Cas continu

Pour un espace d'états continu, les probabilités de transition point par point seront généralement nulles. En conséquence, les transitions seront habituellement exprimées par

des distributions conditionnelles $P(x,A) := \mathbb{P}(X_{t+1} \in A | X_t = x), t \geq 0$, et la distribution de transition en k étapes sera notée $P^k(x,A) := \mathbb{P}(X_{t+k} \in A | X_t = x), t \geq 0$. De plus, certaines notions précédentes doivent être redéfinies ou généralisées pour être applicables de façon intéressante.

1. On dira qu'une chaîne est ϕ-irréductible s'il existe une mesure ϕ non-nulle telle que pour tout ensemble $A \subset S$ avec $\phi(A) > 0$ et pour tout $x \in S$, il existe $n \in \mathbb{N}$ tel que $P^n(x,A) > 0$.
2. On dira qu'une chaîne est périodique s'il existe une suite d'ensembles de mesure non-nulle $A_1, \ldots, A_k$, avec $k \geq 2$, telle que pour $i = 1, \ldots, k-1$,

   $$P(x, A_{i+1}) = 1, \forall x \in A_i,$$

   et

   $$P(x,A_1) = 1, \forall x \in A_k.$$

   Autrement, on dira que la chaîne est apériodique.
3. Une distribution $\pi(\cdot)$ sera dite stationnaire si

   $$\int_{x \in S} \pi(x) P(x,y) \mathrm{d}x = \pi(y).$$

   Notons que par un abus de notation fréquent dans le domaine, les notations $\pi(\cdot)$ et $P(x,\cdot)$ représenteront, selon le contexte, soit les distributions déjà définies, soit les densités induites par ces distributions.
4. Finalement, on dira qu'une chaîne ϕ-irréductible avec distribution stationnaire $\pi(\cdot)$ est Harris-récurrente si pour tout ensemble $A \subset S$ tel que $\pi(A) > 0$ et pour tout $x \in S$,

   $$\mathbb{P}(\exists n \in \mathbb{N} : X_n \in A | X_0 = x) = 1.$$

Ces notions permettent d'énoncer un théorème ergodique pour les chaînes de Markov sur des espaces continus, ainsi qu'une loi des grands nombres sous ces mêmes conditions (voir respectivement [27] et [23] pour une démonstration détaillée). Comme nous le verrons, ces résultats sont essentiels pour les développements qui vont suivre.

Théorème 1.1.1. *Une chaîne de Markov ϕ-irréductible, apériodique et Harris-récurrente de distribution stationnaire $\pi(\cdot)$ convergera en distribution vers cette dernière peu importe la valeur initiale. Ainsi, pour tout $x \in S$:*

$$P^n(x,\cdot) \xrightarrow[n \to \infty]{D} \pi(\cdot).$$

Théorème 1.1.2. *Sous les mêmes conditions, soit une fonction $h : S \to \mathbb{R}$ telle que $\mathbb{E}_\pi[|h(X)|] \in \mathbb{R}$. Alors,*

$$\frac{1}{N} \sum_{n=1}^{N} h(X_n) \xrightarrow[N \to \infty]{p.s.} \mathbb{E}_\pi[h(X)] =: \pi(h),$$

où p.s. dénote la convergence presque sûre.

1.2. Méthodes de Monte-Carlo

Les méthodes de Monte-Carlo désignent de façon générale toute technique de calcul utilisant des procédés aléatoires ou pseudo-aléatoires. Dans un contexte statistique, on s'intéresse à un sous-ensemble de ces méthodes, soit les techniques d'échantillonnage aléatoire numérique visant à calculer des intégrales en plusieurs dimensions. Le problème fondamental considéré est le suivant. Soit X, une variable aléatoire de support $S \subset \mathbb{R}^d$ et de densité $f(x)$, et h, une fonction quelconque définie sur S. On cherche à calculer une bonne approximation de :

$$I = \mathbb{E}_f[h(X)] = \int_S h(x)f(x)\mathrm{d}x.$$

Pour ce faire, on génère un certain nombre n de réalisations $x_i \in S, i = 1, \ldots, n$, indépendantes et identiquement distribuées (i.i.d.) selon la distribution de X, puis on estime I par :

$$\overline{h}_n = \frac{1}{n}\sum_{i=1}^{n} h(x_i).$$

1.2.1. Propriétés

1. Par la loi des grands nombres, avec probabilité 1 :

$$\overline{h}_n \xrightarrow[n\to\infty]{p.s.} \mathbb{E}_f[h(X)].$$

2. À la condition que $\mathbb{E}_f[h^2(X)]$ soit finie :

$$\mathbb{V}(\overline{h}_n) = \frac{1}{n}\mathbb{V}(h(X)) = \frac{1}{n}\left(\int h^2(x)f(x)\mathrm{d}x - \mathbb{E}_f^2[h(X)]\right).$$

3. Sous la même condition, par le théorème limite central :

$$\sqrt{n}\frac{\overline{h}_n - I}{\sqrt{\mathbb{V}(h(X))}} \xrightarrow[n\to\infty]{D} \mathcal{N}(0,1).$$

Ces techniques sont surtout utiles dans les problèmes en grande dimension, où les méthodes numériques traditionnelles perdent de leur efficacité. La difficulté dans l'application de ces méthodes est de trouver une façon de générer efficacement un échantillon de variables i.i.d. de densité f.

1.3. Analyse bayésienne

Dans le cadre du modèle bayésien, nous supposons que les paramètres du modèle à l'étude, que nous dénoterons par le vecteur θ, sont des variables aléatoires suivant une certaine distribution. À partir d'une information *a priori* sur les paramètres, qui prendra la forme d'une distribution de probabilité de densité $p(\theta)$, ainsi qu'un ensemble de données ou d'observations

y provenant du modèle, on déduira une distribution *a posteriori* $p(\theta|y)$. Nous supposerons pour la suite que toutes ces lois sont continues. Par les identités probabilistes de base, on montre que

$$\pi(\theta) := p(\theta|y) = \frac{p(\theta,y)}{p(y)} = \frac{p(y|\theta)p(\theta)}{\int_\Theta p(y|\theta)p(\theta)d\theta},\ \theta \in \Theta,$$

où Θ représente le support des paramètres. Souvent, ce calcul s'avérera problématique. En effet, dans des modèles complexes, il sera impossible d'obtenir une forme explicite ou suffisamment simple pour $p(\theta|y)$. Pour calculer des probabilités et des statistiques de cette distribution, souvent de grande dimension, il faudra pourtant trouver de bonnes estimations d'intégrales de la forme

$$\int h(\theta)\pi(\theta)d\theta.$$

La méthode de Monte-Carlo vue précédemment semble tout indiquée pour répondre à ce problème. Il faudra toutefois être en mesure de simuler des observations de la densité $\pi(\theta)$. Comme nous allons le voir dans le prochain chapitre, les MCMC permettent de répondre de façon très générale à cette dernière problématique.

Chapitre 2

MÉTHODES MCMC

2.1. Introduction

Les méthodes de Monte-Carlo par chaînes de Markov permettent d'élargir grandement l'éventail des distributions pouvant être simulées numériquement. Elles sont relativement simples à implémenter et ne requièrent souvent que la connaissance de la fonction de densité cible à une constante près, ce qui les rend intéressantes dans de nombreuses situations. Cependant, une implémentation naïve peut mener à des temps de calcul très longs, puisque la convergence de ces méthodes est relativement lente lorsqu'elles ne sont pas bien calibrées à une situation donnée.

Nous verrons d'abord les justifications théoriques de ces méthodes, puis nous les illustrerons par la présentation de l'algorithme original de Metropolis-Hastings et de ses propriétés (section 2.2), pour ensuite voir les améliorations successives que l'on peut y apporter, leurs particularités et leur validité théorique. Nous nous concentrerons surtout sur l'ajout d'une composante adaptative au modèle et ses conséquences (section 2.3), posant ainsi les bases du sujet central de ce mémoire. Nous introduirons ensuite l'utilisation de chaînes MCMC parallèles, un procédé souvent utilisé en conjonction avec les algorithmes adaptatifs (section 2.4), et nous conclurons ce chapitre par une description sommaire d'autres techniques et algorithmes d'intérêt, avec les référence appropriées (section 2.5).

L'idée fondamentale des MCMC est de simuler des observations d'une distribution de densité π donnée en utilisant une chaîne de Markov ergodique $\{X_t\}_{t\geq 0}$ dont la distribution stationnaire est π. Le théorème ergodique garantit alors la convergence en loi de $\{X_t\}$ vers une variable de densité f et par conséquent, pour presque toute valeur initiale X_0 :

$$\frac{1}{n}\sum_{t=1}^{n} h(X_t) \xrightarrow[n\to\infty]{p.s.} \mathbb{E}_f[h(X)].$$

Ainsi, nous appellerons MCMC toute méthode permettant de générer des variables d'une distribution en utilisant une chaîne de Markov ergodique ayant celle-ci comme distribution stationnaire. Pour construire un tel algorithme, il faut donc déterminer un noyau de transition

$P(x,\cdot)$ approprié, c'est-à-dire respectant les conditions du théorème ergodique et ayant la bonne distribution stationnaire. Pour remplir cette dernière condition, le résultat suivant s'avérera utile.

Proposition 2.1.1. *Soit une chaîne de Markov ayant un noyau de transition $P(x,\cdot)$ et une distribution $\pi(\cdot)$ définis sur le même espace d'états S. Si P possède la propriété de réversibilité par rapport à π, c'est-à-dire si :*

$$\forall x,y \in S : \pi(x)P(x,y) = \pi(y)P(y,x), \tag{2.1.1}$$

alors la distribution stationnaire de la chaîne est π.

Démonstration. On aura stationnarité si, $\forall y \in S$:

$$\int_S \pi(x)P(x,y)\mathrm{d}x = \pi(y).$$

Or, sous l'hypothèse de réversibilité :

$$\forall y : \int_S \pi(x)P(x,y)\mathrm{d}x = \int_S \pi(y)P(y,x)\mathrm{d}x = \pi(y)\int_S P(y,x)\mathrm{d}x = \pi(y).$$

□

La réversibilité est donc une propriété plus forte que la stationnarité, mais elle est plus simple à vérifier et à manipuler dans notre contexte. En effet, celle-ci permet de construire des probabilités de transition appropriées pour une distribution stationnaire souhaitée.

2.2. L'algorithme de Metropolis-Hastings

Voyons d'abord les détails de l'algorithme dans sa forme générale telle que décrite par [19], pour ensuite montrer sa validité. Pour une densité cible $\pi(\cdot)$ donnée, l'algorithme ne nécessite qu'une valeur de départ X_0 et le choix d'une distribution conditionnelle de densité $q(x,y) := q(y|x)$. Cette dernière servira à proposer, basé sur l'état actuel de la chaîne $\{X_t\}$, un candidat pour l'état suivant de la chaîne. Ce candidat sera ensuite accepté ou rejeté avec une certaine probabilité, choisie de façon à ce que la chaîne soit réversible par rapport à $\pi(\cdot)$. Précisément, à une étape donnée $t \in \mathbb{N}$, les manipulations suivantes sont effectuées.

2.2.1. Algorithme MH

1. À partir de la valeur $X_t = x$, générer la valeur candidate $Y_{t+1} = y$ selon la distribution de densité $q(y,x)$.
2. On pose $X_{t+1} = \begin{cases} Y_{t+1} & \text{avec probabilité} \quad \alpha(x,y), \\ X_t & \text{avec probabilité} \quad 1-\alpha(x,y); \end{cases}$

où le seuil d'acceptation $\alpha(x,y)$ doit avoir la forme générale

$$\alpha(x,y) = \frac{s(x,y)}{1+\frac{\pi(x)q(x,y)}{\pi(y)q(y,x)}}$$

et où la fonction s doit être telle que $s(x,y) = s(y,x)$ et $0 \leq \alpha(x,y) \leq 1$.

Habituellement, on utilise exclusivement la forme

$$\alpha(x,y) = \min\left\{1, \frac{\pi(y)q(x,y)}{\pi(x)q(y,x)}\right\}, \tag{2.2.1}$$

qui correspond au choix $s(x,y) = \min\{1 + \frac{\pi(x)q(x,y)}{\pi(y)q(y,x)}, 1 + \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}\}$. Si de plus la densité q est symétrique ($q(y,x) = q(x,y)$), le rapport devient tout simplement

$$\alpha(x,y) = \min\left\{1, \frac{\pi(y)}{\pi(x)}\right\}. \tag{2.2.2}$$

En particulier, si la densité q est centrée à l'état actuel de la chaîne, de façon à ce que $q(x,y) = q(|y - x|)$, l'algorithme prend alors la forme suivante :

1. À partir de la valeur $X_t = x$, générer la valeur candidate $Y_{t+1} = y$ selon la distribution de densité $q(y,x)$.
2. Calculer $\alpha(x,y) = \min\left\{1, \frac{\pi(y)}{\pi(x)}\right\}$.
3. Choisir $X_{t+1} = \begin{cases} Y_{t+1} & \text{avec probabilité} \quad \alpha(x,y), \\ X_t & \text{avec probabilité} \quad 1 - \alpha(x,y). \end{cases}$

Cette dernière forme est intéressante à interpréter d'un point de vue intuitif. À chaque étape, la quantité $\alpha(x,y)$ compare le candidat Y_{t+1} à l'état actuel de la chaîne X_t. Si Y_{t+1} est plus vraisemblable que X_t du point de vue de la densité $\pi(\cdot)$, donc si $\pi(Y_{t+1}) \geq \pi(X_t)$, Y_{t+1} sera directement accepté. Sinon, il sera accepté seulement avec une certaine probabilité proportionnelle au rapport de densité α. Ainsi, plus un candidat semble représentatif de la distribution $\pi(\cdot)$ par rapport à l'état actuel de la chaîne, plus il aura de chance d'être accepté comme valeur suivante. Si sa densité est trop faible, l'algorithme préférera souvent maintenir l'état actuel de la chaîne, qui est plus représentatif de la distribution $\pi(\cdot)$.

Nous nommerons q la densité de proposition des candidats, ou densité instrumentale, $\alpha(x,y)$ la probabilité d'acceptation ou le seuil d'acceptation, et π la densité cible. Voyons maintenant les propriétés théoriques de cette procédure.

2.2.2. Propriétés

L'algorithme tel que défini génère une chaîne de Markov dont le noyau de transition est donné par :

$$P(x,y) = q(x,y)\alpha(x,y), \text{ si } y \neq x,$$
$$P(x,x) = 1 - \int_{S\setminus\{x\}} P(x,y)\mathrm{d}y.$$

Pour prouver que la distribution stationnaire de cette chaîne est π, par la proposition 2.1.1, il suffit de montrer qu'elle est réversible par rapport à π (voir équation (2.1.1)). Or,

$$\begin{aligned}
\pi(x)P(x,y) &= \pi(x)q(x,y)\alpha(x,y) \\
&= \frac{\pi(x)q(x,y)s(x,y)}{1+\frac{\pi(x)q(x,y)}{\pi(y)q(y,x)}} \\
&= \frac{\pi(x)\pi(y)q(x,y)q(y,x)s(x,y)}{\pi(y)q(y,x)+\pi(x)q(x,y)} \\
&= \frac{\pi(y)q(y,x)s(y,x)}{\frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}+1} \\
&= \pi(y)q(y,x)\alpha(y,x) = \pi(y)P(y,x).
\end{aligned}$$

Ainsi, la chaîne de Markov générée possède bien la distribution stationnaire souhaitée. Maintenant, il faut s'assurer que celle-ci converge bien vers sa distribution stationnaire. Or, ceci est facilement vérifié la plupart du temps.

Considérons, par exemple, le cas où S est un support discret, avec le choix habituel de $\alpha(x,y)$. Si $q(x,y)$ est choisie de façon à être strictement positive pour toute paire $(x,y) \in S^2$, $P(x,y)$ le sera aussi. Ainsi, à partir d'une valeur X_t donnée, toute valeur X_{t+1} sera atteignable en une seule étape avec une probabilité strictement positive, donnant ainsi une chaîne irréductible. Cette dernière sera aussi apériodique du moment qu'il existe au moins une paire (x,y) telle que $\alpha(x,y) < 1$, car on aura alors $P(x,x) > 0$. Cela sera pratiquement toujours vrai et la convergence est alors assurée.

D'autre part, dans le cas qui nous intéressera essentiellement à partir de maintenant, celui où $S \subset \mathbb{R}^d$ pour un certain $d \in \mathbb{N}$, on peut montrer (voir [27]) que si la densité de proposition $q(\cdot,\cdot)$ est continue et strictement positive et que $\pi(\cdot)$ est de mesure finie, alors la chaîne ainsi obtenue convergera vers $\pi(\cdot)$ en distribution. En particulier, ce sera le cas pour n'importe quelle densité de proposition normale.

2.2.3. Remarques

1. C'est le choix de la distribution des candidats $q(x,y)$ qui influencera la qualité de l'algorithme. En effet, un choix judicieux favorisera une exploration rapide de l'espace d'états et accélérera en conséquence la convergence de la chaîne vers la distribution stationnaire. On cherche une variabilité modérée dans les candidats possibles, de façon à favoriser les fluctuations de la chaîne tout en restreignant la probabilité de rejet. On verra dans les sections suivantes plusieurs façons d'ajuster la distribution instrumentale q pour une distribution cible donnée.

   Pour des raisons de performance, on choisira évidemment une distribution q facile à simuler numériquement. En pratique, sur le support $\mathbb{R}^d$, ce sera souvent une loi

normale centrée à la valeur actuelle de la chaîne : $Y_{t+1} \sim \mathcal{N}(X_t,\Sigma)$. C'est donc par le choix de la matrice de covariance Σ que l'on pourra optimiser les performances de l'algorithme. Notons que ce choix rend q symétrique et positive sur S et garantit donc la convergence vers la distribution $\pi(\cdot)$, du moment que cette dernière est de mesure finie.

2. Théoriquement, pour obtenir un véritable échantillon i.i.d. de n observations de la distribution cible, il faudrait générer n chaînes de Markov suffisamment longues et prendre la dernière valeur de chacune. En pratique, on utilise habituellement une seule longue chaîne de laquelle on élimine les B premières valeurs, avec B relativement grand, de façon à conserver uniquement la portion de la chaîne suffisamment proche de la cible. De même, pour diminuer l'impact de la corrélation entre des valeurs consécutives de l'échantillon, on peut choisir de conserver uniquement une valeur sur 10 par exemple pour l'échantillon final.

3. Dans l'algorithme de Metropolis-Hastings standard, puisque les probabilités $\alpha(x,y)$ ne dépendent de la distribution cible que sous la forme du rapport $\pi(y)/\pi(x)$, il est souvent possible d'appliquer l'algorithme sans connaître la constante de normalisation de la densité $\pi(\cdot)$, ce qui est très intéressant dans un contexte d'inférence bayésienne, où cette constante est souvent impossible à déterminer.

2.3. Algorithmes adaptatifs

Comme nous l'avons déjà mentionné, la vitesse de convergence des MCMC est fortement influencée par le choix et la paramétrisation de la distribution q des candidats. Il existe certains critères permettant d'optimiser celle-ci, mais ils dépendent tous d'une certaine connaissance de la covariance de la distribution cible. En effet, il semble intuitivement logique qu'une distribution de candidats appropriée devrait avoir une forme et une échelle semblables à celles de la distribution cible. Or, pour estimer cette information manquante, un échantillon de la distribution cible est nécessaire, qui devra lui aussi provenir d'un algorithme MCMC, forcément non optimal. Pour pallier à cette problématique, on a introduit de nouveaux algorithmes, dits adaptatifs, utilisant une distribution candidate qui évolue au fil des itérations, habituellement en fonction de l'information contenue dans les valeurs X_t déjà générées. Il va de soi que la suite $\{X_t\}$ ainsi obtenue n'est pas une chaîne de Markov, puisque la valeur de X_{t+1} dépendra de toutes les valeurs $X_0,\ldots,X_t$. Tout de même, sous certaines conditions que nous verrons plus loin, l'ergodicité du processus est préservée. Présentons d'abord l'algorithme adaptatif AM (*Adaptive Metropolis* de [16]), très utilisé, largement inspiré de celui de Metropolis-Hastings et respectant les conditions susmentionnées.

2.3.1. Algorithme AM

On considère une distribution cible de densité $\pi(\cdot)$ avec support $S \subset \mathbb{R}^d$, avec la condition supplémentaire que S soit compact. Pour débuter on choisit une valeur initiale $X_0 \in S$, comme auparavant, une matrice $C_0 \in \mathbb{R}^{d\times d}$ symétrique définie positive et un indice $t_0 \in \mathbb{N}$. À chaque étape t, les procédures suivantes sont effectuées.

1. Générer la valeur candidate Y_t à partir d'une distribution normale de moyenne X_{t-1} et de covariance C_t.
2. Comme dans l'algorithme MH, accepter comme valeur pour X_t le candidat Y_t avec probabilité
$$\alpha(X_{t-1}, Y_t) = \min\left(1, \frac{\pi(Y_t)}{\pi(X_{t-1})}\right).$$
Autrement, $X_t := X_{t-1}$.
3. Calculer la matrice de covariance C_{t+1} pour l'étape suivante.

Après le temps t_0 dit de pré-adaptation, C_t sera redéfinie à chaque étape comme la covariance de tous les éléments précédents de la chaîne, ainsi :

$$C_t = \begin{cases} C_0 & \text{pour } t \leq t_0 \\ s_d\left(\mathrm{Cov}(X_0, \ldots, X_{t-1}) + \epsilon I_d\right) & \text{pour } t > t_0, \end{cases} \tag{2.3.1}$$

où s_d est un paramètre constant dépendant uniquement de d, $\epsilon > 0$ est petit, et $\mathrm{Cov}(X_0, \ldots, X_{t-1})$ est la matrice de covariance empirique des vecteurs X_0 à X_{t-1}. Rappelons que

$$\mathrm{Cov}(x_0, \ldots, x_k) := \frac{1}{k}\sum_{i=0}^{k}(x_i - \overline{x}_k)(x_i - \overline{x}_k)^T = \frac{1}{k}\left(\sum_{i=0}^{k} x_i x_i^T - (k+1)\overline{x}_k\overline{x}_k^T\right), \tag{2.3.2}$$

où

$$\overline{x}_k := \frac{1}{k+1}\sum_{i=0}^{k} x_i.$$

Ainsi, ce nouvel algorithme cherche à optimiser la distribution q en lui attribuant comme matrice de covariance une certaine fonction de l'estimé de la matrice de covariance de la cible. Les éléments de la chaîne générés jusqu'à ce point formant en principe un échantillon de plus en plus représentatif de la cible, il est naturel de les utiliser afin de construire cet estimé, qui lui aussi devrait s'améliorer au fil des étapes.

2.3.2. Remarques

1. La raison d'être du second terme de C_t est essentiellement théorique. Il assure que la matrice de covariance restera définie positive et permet de justifier la convergence du processus. En pratique, on peut sans risque choisir $\epsilon = 0$. Parallèlement, la condition de compacité sur S nous assure que les coefficients de C_t sont toujours bornés.

2. Pour une distribution cible normale et une distribution instrumentale normale également, il a été montré que, sous certaines hypothèse supplémentaires, $s_d \approx 2{,}4^2/d$ est le paramètre d'échelle optimal (voir [10] et [26]) et celui-ci est largement utilisé en pratique.

3. La mise à jour de la matrice de covariance peut se faire à l'aide des formules de récurrence suivantes :

$$\overline{x}_t = \frac{t}{t+1}\overline{x}_{t-1} + \frac{x_t}{t+1}, \tag{2.3.3}$$

$$C_{t+1} = \frac{t-1}{t}C_t + \frac{s_d}{t}\left(t\overline{x}_{t-1}\overline{x}_{t-1}^T - (t+1)\overline{x}_t\overline{x}_t^T + x_t x_t^T + \epsilon I_d\right). \tag{2.3.4}$$

   Ainsi, l'ordre de complexité du calcul de C_t est indépendant de t et ne dépend que de la dimension d.

4. La condition de compacité de S n'est pas vraiment contraignante car cela n'empêche pas S d'être très grand, par exemple $S = [-10^5,10^5]^d$. Ainsi, on pourra remplacer une densité cible f^* définie sur $\mathbb{R}^d$ par :

$$f(x) := \begin{cases} f^*(x) & \text{si } x \in S, \\ 0 & \text{autrement.} \end{cases}$$

   Cette modification ne demande aucun calcul supplémentaire (la densité f n'est pas normalisée, mais comme nous l'avons vu, cela n'a pas d'importance quand q est symétrique) et en pratique ne changera en rien les résultats obtenus par l'algorithme. Pour la suite, nous supposerons donc toujours que S est compact, sans que cela ne soit toujours explicitement mentionné.

5. Le temps de pré-adaptation t_0 représente le nombre d'itérations avant que l'adaptation ne débute. Le choix de cette valeur dépendra de la dimension de la cible et du niveau de confiance que l'on a en la matrice de covariance initiale C_0. D'une part, une grande valeur t_0 retardera bien sûr l'effet du processus d'adaptation, mais d'autre part, une valeur trop petite pourrait mener à une sous-exploration de l'espace et donc à une mauvaise première estimation de C_t.

6. Du point de vue de la performance de l'algorithme, il est certain que le temps requis par itération sera plus grand que pour l'algorithme de Metropolis-Hastings original. Toutefois, la meilleure calibration des paramètres obtenue par la version adaptative fera souvent en sorte de réduire grandement le nombre d'itérations requis pour que la chaîne se rapproche suffisamment de la distribution cible, menant ainsi à un gain de temps réel. En fait, dans certaines situations pratiques, par exemple en présence de composantes fortement corrélées, il est essentiellement impossible pour un algorithme sans adaptation de converger en un temps raisonnable. Même dans les situations où

une adaptation ne s'avèrerait pas nécessaire, l'étape adaptative occasionne souvent une charge de calcul marginale comparativement à l'évaluation du rapport de densité.

7. Il existe aussi une version de l'algorithme avec mise à jour composante par composante (voir [17]), qui ne sera pas présentée ici.

2.3.3. Considérations théoriques

Tel que mentionné auparavant, l'ergodicité des algorithmes adaptatifs n'est pas automatique. Dans la plupart des cas, la construction de $\alpha(x,y)$ suffit à préserver la stationnarité par rapport à $\pi(\cdot)$, mais il n'est plus possible d'utiliser directement la théorie des chaînes de Markov pour assurer la convergence vers la stationnarité. Pour montrer l'ergodicité de tels algorithmes, une paire de conditions suffisantes relativement faciles à évaluer a donc été élaborée. Pour énoncer celles-ci, nous utiliserons la notion de distance en variation totale entre deux distributions $P(\cdot)$ et $Q(\cdot)$, prenant valeur sur un support S, que nous noterons par $\|P - Q\|$ et qui se définit ainsi :

$$\|P - Q\| = \sup_A |P(A) - Q(A)|,\ A \subseteq S. \tag{2.3.5}$$

Autrement dit, il s'agit de la plus grande différence de probabilités possible que ces deux distributions peuvent assigner à un même événement. On peut vérifier que cette mesure possède bien toutes les propriétés habituelles de la distance, ainsi que plusieurs autres. En particulier, on peut définir l'ergodicité en utilisant cette notion (voir [27]) : un processus stochastique $\{X_t\}_{t\in\mathbb{N}}$ de distribution stationnaire $\pi(\cdot)$ sera ergodique si, pour toute valeur de X_0 :

$$\|P^t(X_0,\cdot) - \pi(\cdot)\| \xrightarrow[t\to\infty]{} 0.$$

Pour la suite, nous dénoterons par $\Gamma_t, t \in \mathbb{N}$, la variable aléatoire représentant le noyau de transition entre X_t et X_{t+1}, par $\mathcal{G}$ l'espace des distributions de noyaux de transition possibles (donc le support des Γ_t), et par γ une réalisation quelconque de Γ_t. Les seuils $\alpha(x,y)$ étant complètement déterminés par $\pi(\cdot)$ et $q(x,y)$, $\mathcal{G}$ est équivalent à l'espace des distributions de candidat, lui-même équivalent à l'espace des paramètres de la famille des distributions candidates. Par exemple, dans le cas de l'algorithme AM, $\mathcal{G}$ pourrait être représenté par un certain sous-espace des matrices définies positives en d dimensions. Nous utiliserons les notations augmentées suivantes pour les probabilités de transition en un pas et en T pas, respectivement.

$$P_\gamma(x,B) := \mathbb{P}\left(X_{t+1} \in B | X_t = x, \Gamma_t = \gamma\right),$$
$$P_\gamma^T(x,B) := \mathbb{P}\left(X_{t+T} \in B | X_t = x, \Gamma_t = \gamma\right).$$

Nous pouvons maintenant énoncer quelques théorèmes provenant de [28] et dont on pourra trouver les démonstrations dans le chapitre 4.2.

Théorème 2.3.1 (Ergodicité des algorithmes adaptatifs)**.** *Soit un algorithme adaptatif générant un processus stochastique X_t, $t \in \mathbb{N}$, sur un espace d'états S compact, avec un espace de distributions de transitions $\mathcal{G}$ et une distribution stationnaire $\pi(\cdot)$, cette dernière étant identique pour toute distribution de transition $P_\gamma, \gamma \in \mathcal{G}$. Supposons que les deux conditions suivantes soient respectées :*

1. *Ergodicité uniforme simultanée : Pour tout $\epsilon > 0$, on peut trouver $T \in \mathbb{N}$ tel que*

$$\|P_\gamma^{(T)}(x,\cdot) - \pi(\cdot)\| < \epsilon,\ \forall x \in S, \forall \gamma \in \mathcal{G}. \tag{2.3.6}$$

2. *Adaptation déclinante :*

$$D_t := \sup_x \|P_{\Gamma_t}(x,\cdot) - P_{\Gamma_{t-1}}(x,\cdot)\| \xrightarrow[t\to\infty]{} 0,\ \textit{en probabilité.} \tag{2.3.7}$$

Alors, le processus X_t obtenu à partir de cet algorithme est ergodique pour $\pi(\cdot)$.

En d'autres mots, la première condition exige que pour toutes les distributions candidates pouvant être obtenues par l'algorithme, une chaîne homogène utilisant cette distribution soit uniformément ergodique. La seconde demande qu'après un certain temps, la probabilité que deux distributions candidates successives soient significativement différentes tende vers zéro.

Théorème 2.3.2 (Loi faible des grands nombres)**.** *Soit $g : S \to \mathbb{R}$, une fonction bornée et mesurable, et X_i, une chaîne de Markov générée à partir d'un algorithme adaptatif. Alors, sous les mêmes conditions d'ergodicité uniforme simultanée et d'adaptation déclinante et pour toutes valeurs initiales, nous avons :*

$$\frac{\sum_{i=1}^n g(X_i)}{n} \xrightarrow[n\to\infty]{p} \mathbb{E}_\pi[g(X)].$$

Théorème 2.3.3. *L'algorithme AM tel que présenté respecte les conditions (2.3.6) et (2.3.7) et produit donc une chaîne ergodique.*

Les algorithmes adaptatifs sont des outils à la fois puissants et versatiles, mais ils ont aussi leurs limites. Par exemple, dans certains contextes, ils demeurent sensibles à un mauvais choix de paramètres initiaux. Nous présentons maintenant une technique pouvant être intégrée à n'importe quel algorithme déjà existant et qui permet de minimiser ce problème.

2.4. Chaînes parallèles

Dans leur forme originale, les MCMC, de nature essentiellement séquentielle, ne profitent pas particulièrement d'un traitement informatique en parallèle. En effet, à cause du temps de convergence, il n'est habituellement pas avantageux de générer plusieurs courtes chaînes au lieu d'une seule chaîne plus longue. Toutefois, un tel traitement pourrait être souhaitable, voire nécessaire, dans l'une ou l'autre des situations suivantes.

1. Lorsque la distribution cible est multimodale. Dans ce cas, il est fort probable qu'une unique chaîne reste très longtemps coincée dans un seul des modes et qu'elle prenne en conséquence un très grand nombre d'itérations avant de représenter adéquatement la distribution cible. L'utilisation de plusieurs chaînes indépendantes avec des points de départ variés augmentera la probabilité que tous les modes soient suffisamment visités. Il existe aussi des façons d'intégrer un certain degré d'adaptation locale au sein même de l'algorithme, comme nous le verrons plus loin.
2. Dans le cadre d'un algorithme adaptatif : on utilisera alors la covariance des données de toutes les chaînes pour mettre à jour la distribution des candidats (qui sera donc identique dans chaque chaîne). Cela permettra d'obtenir plus rapidement une distribution q de qualité. À titre d'exemple, voici une adaptation à chaînes multiples de l'algorithme AM présenté plus haut, tirée de [6].

2.4.1. Algorithme : adaptation interchaîne

1. On génère un certain nombre m de chaînes suivant l'algorithme AM, ayant chacune sa propre valeur de départ et fonctionnant indépendamment des autres pour les t_0 premières itérations.
2. On construit alors la matrice C_t en tenant compte de la covariance entre toutes les valeurs de toutes les chaînes :

$$C_t = s_d \left(\mathrm{Cov}(X_{1,0}, X_{1,1}, \ldots, X_{1,t-1}, X_{2,0}, \ldots, X_{m,t-1}) + \epsilon I_d\right), \text{ pour } t > t_0.$$

3. On poursuit de la même façon en mettant à jour C_t après chaque itération de chaque sous-chaîne, encore une fois à l'aide des formules de récurrence de l'algorithme AM.
4. On considérera alors l'ensemble des valeurs de toutes les chaînes pour notre échantillon.

Nous verrons à la section 4.5 que cette modification préserve l'ergodicité de l'algorithme AM.

2.5. Autres algorithmes et améliorations

Nous survolons ici certaines autres approches pouvant généralement être ajoutées à un algorithme MCMC générique, outre l'adaptation régionale que nous verrons en détail dans la section suivante. Ces autres options ne seront pas étudiées dans le cadre de la présente recherche, mais sont mentionnées par souci de complétude.

2.5.1. Température

La notion de température peut être utilisée dans les algorithmes adaptatifs pour favoriser le déplacement de la chaîne dans toutes les régions de l'espace et, par le fait même, accélérer

la vitesse de convergence. L'idée est de d'abord simuler comme distribution stationnaire une version aplatie de la distribution cible, jusqu'à ce qu'une certaine mesure empirique de convergence soit satisfaite (par exemple, le R de Brooks-Gelman-Rubin, voir chapitre 5). On poursuit alors les itérations avec la distribution instrumentale adaptative la plus récente et avec une nouvelle distribution stationnaire moins aplatie, et ainsi de suite en progressant vers la véritable distribution cible. Cette méthode peut potentiellement accélérer le processus d'adaptation. Voir [6] pour plus de détails.

2.5.2. Rejet différé

Le rejet différé (proposé dans [24]) est une technique consistant à proposer successivement plus d'un candidat par itération. Si le premier candidat est rejeté, on vérifie si le suivant est accepté et ainsi de suite, jusqu'à acceptation de l'un d'entre eux ou le rejet de tous les candidats. En pratique, on se limitera habituellement à deux candidats par itération. Il est justifié de penser que les seconds candidats générés en utilisant l'information du premier candidat rejeté seront de meilleure qualité. En fait, on peut montrer que sous certaines conditions et en grande dimension, le candidat y_2 optimal est choisi de façon symétrique à y_1 par rapport à x, c'est-à-dire :

$$y_2 = x - (y_1 - x) = 2x - y_1,$$

avec probabilité 1 (voir [4]). La combinaison de cette technique avec l'algorithme AM est discutée dans [14].

2.5.3. Autres algorithmes adaptatifs

Parmi les autres algorithmes adaptatifs d'intérêt, citons l'algorithme dit auto-regénératif (*self-regenerative*, SR) de [30], basé sur la génération d'une chaîne auxiliaire, celui d'Atchade et Rosenthal (voir [2]), où la covariance est un multiple de la matrice identité s'adaptant selon les probabilités empiriques d'acceptation obtenues auparavant, ainsi que les nombreux exemples présentés dans [29] et [1].

Chapitre 3

ALGORITHMES AVEC ADAPTATION RÉGIONALE

3.1. Introduction

Comme nous l'avons mentionné précédemment, même les algorithmes adaptatifs les plus performants perdent de leur efficacité lorsque la distribution cible est multimodale, asymétrique ou de façon générale nettement différente de la distribution de proposition. En effet, dans ces cas problématiques, une unique distribution pour les candidats, même optimale, ne peut explorer l'espace de façon satisfaisante. Nous pouvons alors considérer une distribution de proposition pouvant varier selon les diverses régions de l'espace d'états S. Autrement dit, les spécifications de la distribution du candidat au temps $t+1$ varieront selon l'emplacement de la valeur X_t. Pour appréhender cette situation, nous partirons de l'hypothèse que l'espace d'états S puisse être divisé en K régions $S_{01},\ldots,S_{0K}$, dans lesquelles les distributions de proposition normales optimales seraient $Q_{01},\ldots,Q_{0K}$, respectivement. Ainsi, de façon idéale, on générerait le candidat Y_{t+1} selon la distribution

$$Y_{t+1} \sim \sum_{i=1}^{K} \mathbb{I}(X_t \in S_{0i}) Q_{0i}(X_t,\cdot),$$

où $\mathbb{I}(\cdot)$ est la fonction indicatrice. En pratique, il est impossible de délimiter ces régions de façon exacte, mais on peut possiblement les estimer, disons par $S_1,\ldots,S_K$. À partir de ce point, deux approches adaptatives sont envisageables afin de compenser l'erreur d'estimation.

Premièrement, il est possible de minimiser l'impact de cette erreur en ayant recours à une distribution de proposition plus complexe dont les paramètres refléterons en quelque sorte la qualité d'ajustement de la distribution à sa région de définition. Alternativement, on pourrait tenter d'adapter continuellement les délimitations des régions jusqu'à obtenir un partage optimal de l'espace. Le second choix sera souvent plus intéressant en terme du nombre d'itérations nécessaires pour obtenir un résultat intéressant, mais chacune de ces itérations demandera davantage de calculs et par conséquent, il n'est pas toujours facile de déterminer

laquelle des deux options est préférable en temps réel. Nous nous pencherons maintenant sur les spécifications de ces deux approches telles qu'utilisées dans les algorithmes RAPT (*Regional adAPTive algorithm*) et RAPTOR (*Regional adAPTive algorithm with Online Recursion*). Nous présenterons ensuite les particularités de notre propre algorithme OPRA (*Online Partitioning of the Regional Adaptive algorithm*).

3.2. Algorithme RAPT

3.2.1. Présentation

Dans le cadre de cet algorithme, l'emplacement des régions $S_1, \ldots, S_K$ est fixé d'avance et on aura pour chacune une distribution adaptative Q_j normale. Aucune des distributions Q_j n'étant parfaite pour une région donnée, on utilise plutôt comme proposition globale Q un mélange des Q_j, différent pour chaque région, et prenant la forme suivante

$$Q(x_t,\cdot) = \sum_{i=1}^{K} \mathbb{I}(x_t \in S_i) \sum_{j=1}^{K} \lambda_{ij}^{(t)} Q_j(x_t,\cdot), \tag{3.2.1}$$

où $\sum_j \lambda_{ij}^{(t)} = 1, \forall i$. Ainsi, des poids λ_{ij} idéaux indiqueraient à quel point la proposition Q_j est plus appropriée que les autres dans la région S_i. Il va de soi que ces paramètres seront difficilement choisis d'avance et devront donc être adaptés en cours d'algorithme pour garantir une bonne performance. Pour déterminer des valeurs appropriées pour ces poids, on peut utiliser la distance de saut quadratique moyenne accumulée jusqu'au temps actuel t, définie ainsi :

$$d_{ij}^{(t)} = \frac{\sum_{s \in W_{ij}^{(t)}} \|X_{s+1} - X_s\|_2^2}{|W_{ij}^{(t)}|},$$

où $W_{ij}^{(t)}$ est un ensemble contenant les indices de tous les éléments de la région i qui sont suivis d'une proposition de la densité Q_j jusqu'au temps t, $|\cdot|$ appliquée à un ensemble dénote la cardinalité, et $\|\cdot\|_2$ est la norme euclidienne de dimension appropriée. Lors de l'itération $t+1$, une seule des K^2 valeurs $d_{ij}^{(t)}$ sera modifiée, celle correspondant au couple (i,j) tel que $X_t \in S_i$ et $Y_{t+1} \sim Q_j$. On peut facilement mettre celle-ci à jour par la formule récursive suivante :

$$|W_{ij}^{(t+1)}| = |W_{ij}^{(t)}| + 1, \tag{3.2.2}$$

$$d_{ij}^{(t+1)} = \frac{|W_{ij}^{(t)}|}{|W_{ij}^{(t+1)}|} d_{ij}^{(t)} + \frac{\|X_{t+1} - X_t\|_2^2}{|W_{ij}^{(t+1)}|}. \tag{3.2.3}$$

On définit alors

$$\lambda_{ij}^{(t)} = \begin{cases} \frac{d_{ij}^{(t)}}{\sum_{j=1}^{K} d_{ij}^{(t)}}, & \text{si le dénominateur est non-nul ;} \\ 1/K, & \text{autrement.} \end{cases} \tag{3.2.4}$$

Ainsi, la proposition engendrant la meilleure exploration de l'espace dans une région donnée se verra octroyer un poids de sélection plus grand.

Il est aussi possible et habituel d'adapter en même temps les distributions Q_j, que nous noterons alors $Q_j^{(t)}$. Nous nous en tiendrons ici à des distributions normales, il suffira donc d'adapter leur matrice de covariance respective selon le même processus que dans l'algorithme AM, mais en utilisant pour le calcul de la covariance de $Q_1^{(t)}$ uniquement les valeurs provenant de la région 1 jusqu'au temps $t-1$, et ainsi de suite.

Si les frontières sélectionnées pour délimiter les différentes régions sont relativement exactes et qu'un mélange de propositions normales est une bonne approximation de la densité cible, on s'attend à ce que l'algorithme ait de bonnes performances séparément dans chaque région. Pour assurer un meilleur équilibre inter-régional, il est possible d'ajouter une dernière composante adaptative globale au mélange. On obtiendrait alors la distribution instrumentale suivante :

$$Q^{(t)}(x_t,\cdot) = (1-\beta)\sum_{i=1}^{K}\mathbb{I}(x_t \in S_i)\sum_{j=1}^{K}\lambda_{ij}^{(t)}Q_j^{(t)}(x_t,\cdot) + \beta Q_S^{(t)}(x_t,\cdot), \tag{3.2.5}$$

avec $0 < \beta < 1$ constant et la distribution $Q_S^{(t)}$ adaptant sa covariance en utilisant l'ensemble de l'échantillon obtenu jusqu'alors. Dans [13], on suggère la valeur de $\beta = 0.3$, à laquelle nous nous sommes restreints dans le cadre de ce projet. Cet ajout sera surtout important lorsque les différents modes ou composantes de la distribution cible sont séparés par une région de faible densité, donc difficile à traverser.

Finalement, on a pour les seuils d'acceptation les simplifications suivantes, où q représente la densité de Q :

$$\alpha^*(x,y) = \begin{cases} \dfrac{\pi(y)}{\pi(x)}, & \text{si } x \text{ et } y \text{ sont dans la même région ;} \\ \dfrac{\pi(y)\left((1-\beta)\sum_{j=1}^{K}\lambda_{aj}^{(t)}q_j^{(t)}(y,x)+\beta q_S^{(t)}(y,x)\right)}{\pi(x)\left((1-\beta)\sum_{j=1}^{K}\lambda_{bj}^{(t)}q_j^{(t)}(x,y)+\beta q_S^{(t)}(x,y)\right)}, & \text{si } x \in S_b \text{ et } y \in S_a, a \neq b. \end{cases} \tag{3.2.6}$$

Afin de résumer et clarifier le fonctionnement du processus, voici maintenant une description détaillée de l'algorithme avec les paramètres habituels.

3.2.2. Algorithme

Nous présentons la version standard avec densités instrumentales normales centrées à l'état actuel de la chaîne, avec nombre de régions K fixé, pour une densité cible $\pi(\cdot)$ de support $S \subset \mathbb{R}^d$. Pour débuter, on spécifie la valeur initiale $X_0 \in S$ et un temps de pré-adaptation $t_0 \in \mathbb{N}$. Pour les autres paramètres à initialiser, les valeurs habituelles sont $\beta = 0.3$, $\lambda_{ij}^{(0)} = 1/K$, $\forall(i,j)$, $C_1^{(0)} = c_1 I_d, \ldots, C_K^{(0)} = c_K I_d$, $C_S^{(0)} = c_0 I_d$, avec $c_0, \ldots, c_K \in \mathbb{R}^+$, ces constantes étant sélectionnées au meilleur de nos connaissances *a priori* sur la cible

de façon à ce que les distributions normales associées aux matrices couvrent l'ensemble de l'espace d'intérêt dans chacune des régions (et globalement pour c_0).

À l'étape $t+1$:

1. Générer $Y_{t+1} \sim (1-\beta)\sum_{i=1}^{K} \mathbb{I}(X_t \in S_i) \sum_{j=1}^{K} \lambda_{ij}^{(t)} Q_j^{(t)}(X_t,\cdot) + \beta Q_S^{(t)}(X_t,\cdot)$, avec $Q_j^{(t)}(X_t,\cdot) \sim \mathcal{N}\left(X_t, C_j^{(t)}\right)$.
2. Choisir $X_{t+1} = \begin{cases} Y_{t+1} & \text{avec probabilité} \quad \alpha^*(X_t,Y_{t+1}) \\ X_t & \text{avec probabilité} \quad 1-\alpha^*(X_t,Y_{t+1}) \end{cases}$ (voir (3.2.6)).
3. Si $t+1 < t_0$, incrémenter t et repartir de l'étape 1 ; sinon, poursuivre avec l'adaptation (étape 4 et suivantes) :
4. Soit i tel que $X_t \in S_i$ et j tel que $Y_{t+1} \sim Q_j$ (si $Y_{t+1} \sim Q_S^{(t)}$, sauter à l'étape 5), alors $t \in W_{ij}^{(t+1)}$et donc :

$$|W_{ij}^{(t+1)}| = |W_{ij}^{(t)}| + 1,$$
$$d_{ij}^{(t+1)} = \frac{|W_{ij}^{(t)}|}{|W_{ij}^{(t+1)}|} d_{ij}^{(t)} + \frac{||X_{t+1} - X_t||^2}{|W_{ij}^{(t+1)}|}.$$

Pour toute paire $(a,b) \neq (i,j)$:

$$W_{ab}^{(t+1)} = W_{ab}^{(t)},$$
$$d_{ab}^{(t+1)} = d_{ab}^{(t)}.$$

Pour $b = 1, \ldots, K$:

$$\lambda_{ib}^{(t+1)} = \frac{d_{ib}^{(t+1)}}{\sum_{j=1}^{K} d_{ij}^{(t+1)}},$$

et pour toute paire (a,b) telle que $a \neq i$:

$$\lambda_{ab}^{(t+1)} = \lambda_{ab}^{(t)}.$$

5. Calculer l'adaptation $C_S^{(t)}$ et de $C_i^{(t)}$ (voir équations (2.3.3) et (2.3.4) de l'algorithme AM).
6. Incrémenter t.

Finalement, il est possible de montrer que cette procédure respecte bien les deux conditions suffisantes d'ergodicité des algorithmes adaptatifs lorsque la distribution cible est strictement positive et continue et que S est compact. Ceci sera démontré en détail dans la section 4.3.

3.2.3. Considérations pratiques

Souvent, cet algorithme utilisera uniquement deux régions S_1 et S_2. Toutefois, comme nous le verrons plus loin, un algorithme général avec K régions différentes, quoiqu'un peu

plus ardu à mettre en oeuvre, n'est pas beaucoup plus coûteux au niveau informatique, puisque seulement une fraction de l'ensemble des paramètres doit être mise à jour à chaque étape. Il va de soi que le nombre de régions devrait tout de même rester très modéré.

En pratique, il peut arriver que l'une des régions ne soit pas ou très peu visitée durant la période de pré-adaptation. Ceci peut créer certaines complications numériques dans le calcul des paramètres adaptatifs λ_{ij} et C_j, qui peuvent être évitées en ajoutant de légères perturbations (voir l'annexe C pour plus de détails). Alternativement, on pourrait choisir d'allonger la période de pré-adaptation si l'une des régions prédéfinies n'est pas suffisamment visitée.

Malgré la flexibilité de cet algorithme, il demeure un certain risque que l'un des modes ne soit pas localisé durant la période de pré-adaptation, avec pour conséquence une sous-estimation significative de la covariance de la distribution de proposition globale, ce qui rendra d'autant plus difficile l'exploration du mode sous-représenté. Cette problématique peut facilement être contournée en utilisant une version avec chaînes parallèles, dont les valeurs initiales seraient dispersées dans chaque région, en supposant que la partition adoptée soit assez bonne. Des résultats expérimentaux semblent montrer qu'une seule chaîne par région produit déjà une nette amélioration dans la performance, pour un même nombre d'itérations. La difficulté principale dans l'utilisation de l'algorithme est de déterminer avec peu d'information une partition aussi optimale que possible de l'espace.

3.3. Algorithme RAPTOR

3.3.1. Présentation

Une façon de contourner le problème de la partition initiale des régions est de rendre celle-ci adaptative. Pour justifier la motivation de l'algorithme RAPTOR, nous supposerons que la distribution cible est un mélange de lois à K composantes. Si de plus, chacune de ces composantes est relativement similaire à une loi normale, il serait possible de l'estimer ainsi :

$$\tilde{\pi}(x) = \sum_{k=1}^{K} \lambda_k q_k(x) := \sum_{k=1}^{K} \tilde{\pi}_k(x),$$

avec $\sum_{k=1}^{K} \lambda_k = 1$, $\lambda_k > 0$, et où les q_k sont des densités normales avec moyennes et covariances respectives μ_k et C_k, pour $k = 1, \ldots, K$. On imagine facilement que pour des distributions de candidats normales, les régions de la partition idéale seraient très proches de :

$$S_k = \left\{ x \in S \,\middle|\, \arg\max_{1 \leq k \leq K} \tilde{\pi}_k(x) = k \right\}.$$

L'idée derrière l'algorithme est donc d'estimer de façon adaptative la moyenne, la covariance et le poids de chacune des composantes, puis de comparer les densités normales utilisant ces estimés afin de décider à quelle région appartiendrait un point donné.

Pour démarrer, l'algorithme aura donc besoin de valeurs initiales hypothétiques pour les moyennes $\mu_k^{(0)}$, les covariances $\Sigma_k^{(0)}$ et les poids $\lambda_k^{(0)}$ de chaque composante k, desquels on peut déduire une première partition de l'espace.

Le processus étant adaptatif, après une période de pré-adaptation, ces mêmes paramètres seront mis à jour à chaque étape. La distribution du candidat au temps $t+1$ sera alors donnée par :

$$Q^{(t)}(X_t,\cdot) = (1-\beta)\sum_{k=1}^{K} \mathbb{I}\{x \in S_k^{(t)}\} Q_k^{(t)}(X_t,\cdot) + \beta Q_S^{(t)}(X_t,\cdot),$$

où les $Q_k^{(t)}$ sont de loi normale de moyennes et covariances respectives $\mu_k^{(t)}$ et $C_k^{(t)}$ pour $k = 1, \cdots, K$, $0 \leq \beta \leq 1$ et $C_S^{(t)}$ est calculée à partir de la covariance globale, comme dans l'algorithme RAPT. On remarquera que les poids adaptatifs $\lambda_k^{(t)}$ n'apparaissent pas dans la distribution des candidats. D'après [3], il est préférable de ne pas les inclure dans la sélection des candidats, en raison de leur volatilité en début d'algorithme.

La mise à jour des autres paramètres est basée sur l'algorithme EM (introduit originalement par [7]) et sera spécifiée dans la description détaillée de l'algorithme qui suit.

3.3.2. Algorithme

Il faut d'abord spécifier les valeurs initiales $X_0, \mu_1^{(0)}, \ldots, \mu_K^{(0)} \in \mathbb{R}^d; C_1^{(0)}, \ldots, C_K^{(0)} \in \mathbb{R}^{d\times d}, t_0 \in \mathbb{N}$, et initialiser les paramètres suivants à leur valeur habituelle : $\beta = 0.3$, $\lambda_k^{(0)} = 1/K$, $\forall k$, $C_S^{(0)} = c_0 I_d$, avec $c_0 \in \mathbb{R}^+$ sélectionné de façon à ce que $C_S^{(0)}$ couvre l'ensemble de l'espace d'intérêt.

À l'étape $t+1$:

1. Générer $Y_{t+1} \sim (1-\beta)\sum_{i=1}^{K} \mathbb{I}(X_t \in S_i^{(t)}) Q_i^{(t)}(X_t,\cdot) + \beta Q_S^{(t)}(X_t,\cdot)$,
   avec $Q_i^{(t)}(X_t,\cdot) \sim \mathcal{N}\left(X_t, C_i^{(t)}\right)$.
2. Choisir $X_{t+1} = \begin{cases} Y_{t+1} & \text{avec probabilité} \quad \alpha^*(X_t,Y_{t+1}) \\ X_t & \text{avec probabilité} \quad 1-\alpha^*(X_t,Y_{t+1}) \end{cases}$ (voir (3.2.6)).
3. Si $t+1 < t_0$, incrémenter t et repartir de l'étape 1 ; sinon, poursuivre avec l'adaptation à l'étape 4.
4. Mettre à jour les paramètres suivants :

$$v_k^{(t+1)} = \frac{\lambda_k^{(t)} q_k^{(t)}(x_{t+1},\cdot)}{\sum_{k=1}^{K} \lambda_k^{(t)} q_k^{(t)}(x_{t+1},\cdot)},$$

$$\lambda_k^{(t+1)} = \lambda_k^{(t)} + \frac{1}{t+2}(v_k^{(t+1)} - \lambda_k^{(t)}),$$

$$
\begin{aligned}
\gamma_k^{(t+1)} &= \frac{v_k^{(t+1)}}{(t+2)\lambda_k^{(t+1)}}, \\
\mu_k^{(t+1)} &= \mu_k^{(t)}(1-(t+1)^{-0.1}\gamma_k^{(t+1)}) + (t+1)^{-0.1}\gamma_k^{(t+1)} x_{t+1}, \\
C_k^{(t+1)} &= C_k^{(t)}(1-(t+1)^{-0.1}\gamma_k^{(t+1)}) \\
&\quad + (t+1)^{-0.1}\gamma_k^{(t+1)} s_d \left((1-\gamma_k^{(t+1)})(x_{t+1}-\mu_k^{(t)})(x_{t+1}-\mu_k^{(t)})^T + \epsilon I_d\right).
\end{aligned}
$$

5. Calculer l'indice de la région de x_{t+1} :

$$
\arg\max_{1\leq k\leq K} \left\{ q_k^{(t)}(x_{t+1}) \right\}.
$$

6. Calculer l'adaptation de la matrice $C_S^{(t+1)}$ (voir équations (2.3.3) et (2.3.4) de l'algorithme AM). Incrémenter t.

Il est à noter que dans [3], l'exposant affecté à $(t+1)$ à l'étape 4 est plutôt -1.1. Or, un tel choix mène à des résultats peu intéressants, puisque l'adaptation y devient négligeable très rapidement. Un examen de l'implémentation informatique de l'algorithme a révélé certains désaccords par rapport à ce qui est décrit dans la référence. L'algorithme tel que présenté ici a donc été ajusté à la suite de l'examen du programme informatique fourni par les auteurs. Cette version modifiée donne des résultats numériques beaucoup plus comparables à ce qui est rapporté dans la référence et nous avons utilisé celle-ci dans nos tests comparatifs.

3.3.3. Considérations pratiques

En comparant les étapes de cet algorithme à celles du précédent, on réalise que la mise à jour adaptative des paramètres requiert un nombre d'opérations élémentaires comparable pour les deux, le RAPTOR étant légèrement plus demandant. De plus, lors d'une étape où le candidat généré Y_{t+1} serait dans une région différente de celle du point actuel X_t, le nombre de calculs de densités normales serait identique, soit $2K+2$. Par contre, dans le cas beaucoup plus fréquent en pratique où X_t et Y_{t+1} sont dans la même région, l'algorithme RAPT ne calculera aucune densité normale, alors qu'au moins $2K$ densités normales seront calculées par RAPTOR, une différence potentiellement non-négligeable sur des problèmes de grande taille.

Il est possible de montrer que sous des contraintes sur $\pi(\cdot)$ semblables à celles imposées à l'algorithme RAPT, l'algorithme RAPTOR respecte les conditions suffisantes d'ergodicité décrites en (2.3.6) et (2.3.7) (voir [3] pour plus de détails).

3.4. Algorithme OPRA

3.4.1. Présentation

L'algorithme que nous avons développé est une modification de l'algorithme RAPT, applicable surtout dans les cas où l'on souhaite partitionner l'espace en deux régions uniquement, ce qui de toute façon s'avère suffisant dans la plupart des situations. Le principe est toutefois généralisable à plus de deux régions, comme nous le verrons plus loin. L'idée de base est d'utiliser le même processus adaptatif que dans RAPT, de permettre aussi l'adaptation de la partition de l'espace, mais que cette délimitation soit aussi simple que possible à redéfinir. La solution la plus simple serait bien sûr que la frontière soit un hyperplan.

Nous cherchons donc à ajouter au RAPT une composante de complexité minimale qui déterminera de façon adaptative le plan séparant le mieux possible les deux régions théoriquement optimales. Ce choix est motivé par le fait que dans bien des cas, même si la séparation idéale entre deux modes est loin d'être linéaire, en considérant uniquement l'ensemble des points de l'espace ayant une densité moindrement significative, nous pouvons aisément faire passer un plan qui sépare les deux modes de façon presque parfaite. De plus, l'utilisation d'une distribution de proposition multimodale avec poids adaptatifs confère une certaine protection contre une partition sous-optimale et devrait aussi favoriser la diffusion de la chaîne même avant que l'adaptation du plan ne soit terminée. Il sera plus difficile de trouver une bonne frontière linéaire lorsque les deux modes sont si près l'un de l'autre que les distributions se confondent, mais dans une telle situation, une adaptation régionale n'est probablement pas nécessaire, car un algorithme adaptatif simple aurait alors une performance similaire.

Tout cela mène à croire que la plupart du temps, ce nouvel algorithme pourrait avoir une précision plus proche de celle du RAPTOR pour un même nombre d'itérations, tout en étant plus rapide en temps réel. Voyons d'abord les détails du calcul du plan adaptatif.

3.4.2. Description de l'algorithme

Pour construire notre plan optimal à une étape donnée, nous aurons besoin des moyennes échantillonnales dans chaque région empirique. Définissons $W_i^{(t)} = \{0 \leq s \leq t : x_s \in S_i^{(t)}\}$ pour $i = 1{,}2$, des ensembles contenant les indices des valeurs appartenant aux régions $S_1^{(t)}, S_2^{(t)}$. Alors,

$$\hat{\mu}_i^{(t)} = \sum_{s \in W_i^{(t)}} \frac{x_s}{|W_i^{(t)}|}\ , \quad i = 1{,}2.$$

On détermine alors les paramètres du plan d'équation $a_t^T X = b_t$ ainsi :

$$a_t = \hat{\mu}_1^{(t)} - \hat{\mu}_2^{(t)}, \tag{3.4.1}$$

$$b_t = a_t^T \left(\frac{\hat{\mu}_1^{(t)} + \hat{\mu}_2^{(t)}}{2} \right). \tag{3.4.2}$$

Ce plan sera la frontière entre les nouvelles régions $S_1^{(t)}$ et $S_2^{(t)}$. On déterminera l'emplacement d'une nouvelle valeur ainsi :

$$X \in \begin{cases} S_1^{(t)} & \text{si} \quad a_t^T X \geq b_t, \\ S_2^{(t)} & \text{si} \quad a_t^T X < b_t. \end{cases}$$

On définit donc le plan de façon perpendiculaire à la droite joignant les moyennes estimées $\hat{\mu}_1^{(t)}$ et $\hat{\mu}_2^{(t)}$ de chaque région, et passant par le point milieu de cette droite. En fait, ce choix est équivalent à attribuer à une nouvelle observation la région dont la moyenne est la plus proche. Si les deux modes de la distribution cible sont de forme et d'échelle relativement similaires, ce choix sera clairement optimal.

S'il arrivait que l'une des deux régions ne soit pas du tout visitée durant la période de pré-adaptation (ce qui suggère que la partition initiale était inappropriée, ou la pré-adaptation trop courte), alors l'une des moyennes empiriques ne sera pas définie (supposons $\hat{\mu}_2$) et on pourrait alors choisir par exemple un plan passant par l'autre moyenne et d'orientation arbitraire : $a = (1, \ldots, 1)$, $b = a^T \hat{\mu}_1$. En pratique, on évitera cette situation en utilisant des chaînes parallèles ayant des valeurs initiales dispersées de part et d'autre de la délimitation initiale.

Du point de vue informatique, les valeurs initiales nécessaires sont les mêmes que pour l'algorithme RAPT et la mise à jour des données se fera de la même façon, mais avec l'ajout de l'étape suivante entre les étapes 2 et 3 de l'algorithme présenté à la section 3.2.2.

Étape 2.5 : En supposant que $X_{t+1} \in S_1^{(t)}$, l'autre possibilité étant analogue :

$$\begin{aligned} \hat{\mu}_1^{(t+1)} &= \hat{\mu}_1^{(t)} + \frac{x_{t+1} - \hat{\mu}_1^{(t)}}{|W_i^{(t+1)}|}, \\ \hat{\mu}_2^{(t+1)} &= \hat{\mu}_2^{(t)}, \\ a_{t+1} &= \hat{\mu}_1^{(t+1)} - \hat{\mu}_2^{(t+1)}, \\ b_{t+1} &= a_{t+1}^T \left(\frac{\hat{\mu}_1^{(t+1)} + \hat{\mu}_2^{(t+1)}}{2} \right). \end{aligned}$$

De même, bien que d'une étape à l'autre les régions puissent changer, à un temps t fixé, les formes de la distribution instrumentale $Q^{(t)}(x_t, \cdot)$ et du seuil d'acceptation $\alpha^*(x_t, y_{t+1})$ seront les mêmes que pour l'algorithme RAPT (voir (3.2.5) et (3.2.6)).

On pourrait se demander à quel point les poids λ_{ij} demeurent utiles dans ce nouvel algorithme. D'après les expérimentations effectuées jusqu'à présent, leur conservation accélère grandement la découverte du plan optimal, probablement parce qu'ils favorisent une meilleure exploration de l'espace avant que l'adaptation du plan ne soit complétée.

Finalement, puisque les éléments a_t et b_t sont calculés à partir de moyennes et covariances empiriques de points provenant d'un espace compact, on s'attendrait à ce que l'hyperplan se stabilise éventuellement. L'algorithme respecterait alors les conditions suffisantes d'ergodicité déjà énoncées. Nous montrons de façon détaillée dans le chapitre 4 que c'est bien le cas. De plus, le peu d'opérations supplémentaires requises à chaque itération par rapport à l'algorithme RAPT occasionnera des temps de calculs comparables pour les deux procédures. Nous terminons cette présentation en décrivant quelques ajouts et généralisations envisageables pour cet algorithme, dont certains ont été testés en pratique.

3.4.3. Calcul itératif des régions

Remarquons que la formule de mise à jour précédente ne reflète pas tout à fait le raisonnement théorique présenté auparavant, puisqu'à une itération donnée, on n'évalue pas le nouvel emplacement de l'ensemble des points simulés jusqu'alors, mais uniquement celui du plus récent. Or, il est clair que la région d'appartenance de certains points risque de changer en cours d'algorithme. Cependant, on peut s'attendre à ce qu'à long terme, ces valeurs erronées aient une influence négligeable dans le calcul des valeurs adaptatives, et qu'une fois la frontière stabilisée, il serait inefficace de recalculer à chaque étape l'ensemble des emplacements, qui risquent de très peu changer. En fait, une ré-évaluation de l'ensemble des points à chaque itération conduirait à des temps de calculs prohibitifs.

Toutefois, en début d'algorithme, cette manipulation supplémentaire pourrait s'avérer intéressante, en particulier lors de la première adaptation, au temps t_0. On pourra alors déterminer les emplacements une première fois pour le calcul des moyennes $\hat{\mu}$ et des paramètres a et b, puis les évaluer une seconde fois selon le nouveau découpage de $S_1^{(t_0)}$ et $S_2^{(t_0)}$. En fait, toute cette procédure (attribution des régions pour chaque point, calcul des moyennes, définition d'un nouvel hyperplan) peut être répétée de nombreuses fois, jusqu'à stabilisation de l'hyperplan, avant même de poursuivre avec la prochaine étape de simulation. Cela nous assurerait d'avoir une séparation aussi satisfaisante que possible au terme de la période de pré-adaptation. Cette sous-procédure est équivalente à l'algorithme des K-moyennes, avec $K = 2$, dont la convergence est assurée (voir détails en annexe B). La pertinence de cet ajout a été évaluée dans nos tests comparatifs.

Tel que déjà mentionné, il n'est pas efficace d'effectuer un tel calcul à chaque itération. Il pourrait être par contre envisageable de le faire de façon sporadique après la première adaptation, par exemple à toutes les 10^4 ou 10^5 itérations. Afin d'assurer une efficacité maximale, on pourrait cesser ces adaptations lorsque l'hyperplan semble suffisamment stabilisé, c'est-à-dire lorsque la grande majorité des points ne change pas de région au terme du recalcul. Des simulations numériques préliminaires suggèrent que ces calculs supplémentaires n'augmentent pas en général la qualité de l'algorithme. En conséquence, cette piste n'a pas été davantage explorée dans ce projet de recherche.

3.4.4. Ajustement pondéré de l'hyperplan

Par ailleurs, dans certaines situations, il serait juste de penser que le choix de faire passer l'hyperplan à mi-distance entre les deux moyennes empiriques n'est pas optimal, par exemple si la densité cible présente une forme ou une échelle différente d'un mode à l'autre. Dans ces cas, il serait pertinent de tenir compte aussi de la covariance empirique observée dans chaque région afin de choisir une position optimale entre les deux moyennes pour l'hyperplan séparateur. Le paramètre b_t ainsi généralisé serait alors calculé à partir d'une moyenne pondérée de $\hat{\mu}_1^{(t)}$ et $\hat{\mu}_2^{(t)}$.

Une façon naturelle de déterminer cette pondération est basée sur la distance de Mahalanobis, une généralisation multidimensionnelle de la cote Z. Pour une observation $x \in \mathbb{R}^d$ de moyenne μ et de covariance C, elle est définie ainsi :

$$D_M(x) = \sqrt{(x-\mu)^T C^{-1}(x-\mu)}.$$

Autrement dit, $D_M(x)$ mesure la distance en écarts-types entre x et sa moyenne. Il serait alors logique de choisir comme position pour l'hyperplan le point r_t situé sur le segment reliant $\hat{\mu}_1^{(t)}$ et $\hat{\mu}_2^{(t)}$, et équidistant entre ces deux points au sens de Mahalanobis, donc tel que

$$(r_t - \hat{\mu}_1^{(t)})^T (C_1^{(t)})^{-1}(r_t - \hat{\mu}_1^{(t)}) = (r_t - \hat{\mu}_2^{(t)})^T (C_2^{(t)})^{-1}(r_t - \hat{\mu}_2^{(t)}).$$

On aurait alors, pour une certaine valeur $k_t \in (0,1)$:

$$r_t = \hat{\mu}_1^{(t)} + k_t(\hat{\mu}_2^{(t)} - \hat{\mu}_1^{(t)}),$$

l'équation à satisfaire devenant alors par substitution

$$k_t^2(\hat{\mu}_2^{(t)} - \hat{\mu}_1^{(t)})^T (C_1^{(t)})^{-1}(\hat{\mu}_2^{(t)} - \hat{\mu}_1^{(t)}) = (1-k_t)^2(\hat{\mu}_2^{(t)} - \hat{\mu}_1^{(t)})^T (C_2^{(t)})^{-1}(\hat{\mu}_2^{(t)} - \hat{\mu}_1^{(t)}).$$

Si $C_1^{(t)} = C_2^{(t)}$, on a directement $k_t = 1/2$. Sinon, k_t s'obtient en résolvant une forme quadratique et s'exprime ainsi :

$$k_t = \frac{\sqrt{z_1 z_2} - z_2}{z_1 - z_2},$$

où $z_1 = (\hat{\mu}_2^{(t)} - \hat{\mu}_1^{(t)})^T (C_1^{(t)})^{-1}(\hat{\mu}_2^{(t)} - \hat{\mu}_1^{(t)})$ et $z_2 = (\hat{\mu}_2^{(t)} - \hat{\mu}_1^{(t)})^T (C_2^{(t)})^{-1}(\hat{\mu}_2^{(t)} - \hat{\mu}_1^{(t)})$. On obtient donc finalement :

$$b_t = a_t^T r_t = a_t^T \left((1-k_t)\hat{\mu}_1^{(t)} + k_t \hat{\mu}_2^{(t)}\right).$$

Une telle résolution nécessite bien sûr des calculs supplémentaires, mais moins que l'on pourrait le croire à première vue, puisqu'une décomposition des matrices $C_1^{(t)}$ et $C_2^{(t)}$ est déjà requise lors de certaines itérations par n'importe quelle implémentation de l'algorithme (voir annexe C pour plus de détails), ce qui accélère le calcul de formes impliquant $(C_1^{(t)})^{-1}$ et $(C_2^{(t)})^{-1}$. Malgré ces manipulations supplémentaires, nos résultats numériques préliminaires ont montré que les temps de calculs demeurent comparables à ceux de l'algorithme RAPT. Cette version a donc été évaluée dans nos simulations.

3.4.5. Généralisation pour plus de 2 régions

Pour $K > 2$, il y aura $\binom{K}{2}$ hyperplans séparant chaque paire de moyennes échantillonnales. On pourra déterminer la région d'appartenance d'un point donné à l'aide de comparaisons successives. Par exemple, pour $K = 3$, pour une observation x, où $a_{ij}^{(t)}$ et $b_{ij}^{(t)}$ représentent les coefficients respectifs des hyperplans séparant $\hat{\mu}_i^{(t)}$ et $\hat{\mu}_j^{(t)}$, $1 \leq i < j \leq K$, et obtenus par des calculs analogues à (3.4.1) et (3.4.2), nous avons :

1. Si $(a_{12}^{(t)})^T x \geq b_{12}^{(t)}$ et $(a_{13}^{(t)})^T x \geq b_{13}^{(t)}$: $x \in S_1$;
2. Si $(a_{12}^{(t)})^T x < b_{12}^{(t)}$ et $(a_{23}^{(t)})^T x \geq b_{23}^{(t)}$: $x \in S_2$;
3. Sinon, $x \in S_3$.

Cette procédure se généralise très naturellement pour $K > 3$ et il est possible de l'implémenter de façon à ce qu'elle ne nécessite que $K - 1$ produits scalaires et comparaisons pour classer un point donné. Notons également que seulement $K - 1$ hyperplans parmi les $\binom{K}{2}$ doivent être mis à jour à chaque itération, puisqu'une seule moyenne empirique est modifiée à chaque itération.

Comme c'était le cas pour $K = 2$, si l'on fait passer les hyperplans par les points milieux des droites joignant les paires de moyennes, cette méthode est équivalente au fait de classer une observation donnée dans la région dont la moyenne est la plus proche. En effet, il est clair que l'hyperplan entre $\hat{\mu}_1^{(t)}$ et $\hat{\mu}_2^{(t)}$ sépare l'espace avec d'un côté la région des points étant plus proches de $\hat{\mu}_1^{(t)}$ que de $\hat{\mu}_2^{(t)}$, et de l'autre, ceux étant plus proches de $\hat{\mu}_2^{(t)}$ que de $\hat{\mu}_1^{(t)}$, et ainsi de suite pour les autres hyperplans.

En fait, pour le classement d'une seule observation lorsque $K \geq 3$, il sera plus efficace de simplement calculer sa distance par rapport à chacune des moyennes, puis de sélectionner la région dont la moyenne minimise la distance sans calculer les hyperplans. Pour $K = 2$, les deux méthodes ont une efficacité assez semblable. Toutefois, si l'on souhaite classer un grand nombre n de points dans une même itération, par exemple lors d'une première adaptation, il est préférable de déterminer d'abord les hyperplans, du moment que

$$\binom{K}{2} + n < Kn,$$

ce qui sera essentiellement toujours le cas, K étant de taille très modérée. Ici aussi, le fait de ré-évaluer à répétition les frontières des régions puis les emplacements est équivalent à l'algorithme des K-moyennes.

Bien que cette généralisation soit facile à mettre en oeuvre et relativement efficace, rappelons que son intérêt pratique est limité. Nous n'explorerons pas davantage les situations où $K > 2$ dans le cadre de cette recherche.

Chapitre 4

RÉSULTATS THÉORIQUES

Nous verrons dans ce chapitre les résultats détaillés permettant de montrer la convergence vers la distribution cible des algorithmes adaptatifs présentés auparavant. Ces résultats permettront également de démontrer une loi des grands nombres pour ces mêmes processus adaptatifs. Ce travail mènera ultimement à la preuve que le processus stochastique généré par le nouvel algorithme proposé converge bien vers la distribution souhaitée.

Nous débuterons le tout par la démonstration de certaines propriétés de la distance en variation totale, qui seront utilisées dans les preuves suivantes. Nous montrerons ensuite la validité des conditions suffisantes d'ergodicité des algorithmes adaptatifs ((2.3.6) et (2.3.7)) déjà énoncées. Nous utiliserons ensuite ces conditions pour démontrer successivement l'ergodicité des algorithmes AM, RAPT et OPRA, ainsi que celle d'un processus à chaînes parallèles basé sur ces mêmes algorithmes.

4.1. Distance en variation totale

Rappelons d'abord la définition de la distance en variation totale, sur un espace S avec une σ-algèbre $\mathcal{F}$:

$$\|P - Q\| := \sup_{A \in \mathcal{F}} |P(A) - Q(A)|,$$

Rappelons également que la propriété d'ergodicité d'un processus stochastique quelconque (X_t) peut être exprimée à l'aide de cette distance :

$$\|P^t(X_0, \cdot) - \pi(\cdot)\| \xrightarrow[t \to \infty]{} 0.$$

Nous montrons maintenant quelques propriétés utiles pour la suite.

Théorème 4.1.1. *Soient $\mu(\cdot)$ et $\nu(\cdot)$, deux mesures de probabilité sur un même support $\mathcal{X}$ de densités respectives g et h par rapport à une mesure p σ-finie, ainsi que $M := \max(g,h)$ et $m := \min(g,h)$. Alors,*

1.

$$\|\mu(\cdot) - \nu(\cdot)\| = \frac{1}{2} \int_{\mathcal{X}} (M - m)\, dp = 1 - \int_{\mathcal{X}} m\, dp.$$

2. *Pour toutes variables aléatoires conjointes X et Y telles que $X \sim \mu(\cdot)$ et $Y \sim \nu(\cdot)$,*

$$\|\mu(\cdot) - \nu(\cdot)\| \leq \mathbb{P}(X \neq Y). \tag{4.1.1}$$

3. *Il existe une paire de variables aléatoires conjointes X et Y telles que $X \sim \mu(\cdot)$, $Y \sim \nu(\cdot)$ et*

$$\mathbb{P}(X = Y) = 1 - \|\mu(\cdot) - \nu(\cdot)\|, \tag{4.1.2}$$

autrement dit, on peut atteindre l'égalité dans la deuxième propriété avec un choix approprié de X et Y.

Démonstration. (Tirée de [27])

1. Nous montrons successivement les deux égalités.
   (a) Notons que $M - m = (g - h)\mathbb{I}_{(g \geq h)} + (h - g)\mathbb{I}_{(g<h)}$. Ainsi,

   $$\frac{1}{2}\int_{\mathcal{X}}(M - m)\mathrm{d}p = \frac{1}{2}\left(\int_{g>h}(g - h)\mathrm{d}p + \int_{g<h}(h - g)\mathrm{d}p\right) = \frac{1}{2}\left|\int_{\mathcal{X}}(g - h)f\mathrm{d}p\right|,$$

   avec $f = \mathbb{I}(g > h) - \mathbb{I}(g \leq h)$. De plus, puisque $g = d\mu/\mathrm{d}p$, $h = d\nu/\mathrm{d}p$, et en définissant $A = \{x \in \mathcal{X} : g(x) > h(x)\}$:

   $$\begin{aligned}\frac{1}{2}\left|\int_{\mathcal{X}}(g - h)f\mathrm{d}p\right| &= \frac{1}{2}\left|\int_{\mathcal{X}} f d\mu - \int_{\mathcal{X}} f d\nu\right| \\ &= \frac{1}{2}|\mu(A) - \mu(A^c) - (\nu(A) - \nu(A^c))| \\ &= |\mu(A) - \nu(A)| = \|\mu(\cdot) - \nu(\cdot)\|,\end{aligned}$$

   puisque A (ou son complément) est nécessairement l'ensemble maximisant $|\mu(\cdot) - \nu(\cdot)|$.
   (b) Notons que $M + m = g + h$ et donc que

   $$\int_{\mathcal{X}}(M + m)\mathrm{d}p = \int_{\mathcal{X}} g\mathrm{d}p + \int_{\mathcal{X}} h\mathrm{d}p = 2.$$

   Ainsi,

   $$\begin{aligned}\frac{1}{2}\int_{\mathcal{X}}(M - m)\mathrm{d}p &= 1 - \frac{1}{2}\left(2 - \int_{\mathcal{X}}(M - m)\mathrm{d}p\right) \\ &= 1 - \frac{1}{2}\left(\int_{\mathcal{X}}(M + m)\mathrm{d}p - \int_{\mathcal{X}}(M - m)\mathrm{d}p\right) \\ &= 1 - \int_{\mathcal{X}} m\mathrm{d}p,\end{aligned}$$

   ce qui montre la seconde égalité.
2. Pour tout ensemble mesurable $A \subset S$:

   $$\begin{aligned}\mathbb{P}(X \in A) &= \mathbb{P}(X \in A, Y \notin A) + \mathbb{P}(X \in A, Y \in A) \\ &\leq \mathbb{P}(X \neq Y) + \mathbb{P}(Y \in A),\end{aligned}$$

impliquant que

$$\mathbb{P}(X \in A) - \mathbb{P}(Y \in A) \leq \mathbb{P}(X \neq Y).$$

On montre de façon analogue que $\mathbb{P}(Y \in A) - \mathbb{P}(X \in A) \leq \mathbb{P}(X \neq Y)$ et donc,

$$|\mathbb{P}(X \in A) - \mathbb{P}(Y \in A)| \leq \mathbb{P}(X \neq Y), \forall A.$$

3. Définissons

$$a := \int_{\mathcal{X}} m \mathrm{d}p, \qquad b := \int_{\mathcal{X}} (g - m) \mathrm{d}p = 1 - a, \qquad c := \int_{\mathcal{X}} (h - m) \mathrm{d}p = 1 - a.$$

Si $a = 0$, alors presque partout, au moins l'une des densité est nulle. Alors, $\mathbb{P}(X = Y) = 0$ pour tout choix de X et Y (puisque leurs supports sont disjoints presque sûrement), et effectivement, $\|\mu(\cdot) - \nu(\cdot)\| = 1$, par la première propriété.

Si $b = 0$ ou $c = 0$, alors les deux vaudront 0 et nous aurons alors $m = g = h$, et donc $\mu(\cdot) = \nu(\cdot)$ presque partout. Dans ce cas, nous aurons forcément $\|\mu(\cdot) - \nu(\cdot)\| = 0$ et en choisissant $Y = X$ comme couplage, nous avons effectivement $\mathbb{P}(X = Y) = 1$.

Si a, b et c sont tous non-nuls, nous construisons les variables aléatoires conjointes Z,U,V,I, telles que Z, U et V aient comme densités respectives m/a,$(g - m)/b$ et $(h - m)/c$, et telles que I soit une indicatrice indépendante des autres variables avec $\mathbb{P}(I = 1) = a$. Nous posons $X = Y = Z$ si $I = 1$ et $X = U$, $Y = V$ sinon. La densité de X est alors donnée par

$$\mathbb{P}(I = 1)(m/a) + \mathbb{P}(I = 0)(g - m)/b = a(m/a) + (1 - a)(g - m)/b = g,$$

tel que souhaité. De même, la densité de Y est bien h. Finalement, U et V ont des supports disjoints, puisque $(g - m) > 0 \Rightarrow h = m \Rightarrow h - m = 0$, et vice-versa. Ainsi, $\mathbb{P}(U = V) = 0$ et donc, en utilisant la première propriété montrée,

$$\mathbb{P}(X = Y) = \mathbb{P}(I = 1) = a = \int_{\mathcal{X}} m \mathrm{d}p = 1 - \|\mu(\cdot) - \nu(\cdot)\|.$$

□

4.2. Ergodicité des algorithmes adaptatifs

Nous pouvons maintenant prouver la validité des conditions suffisantes d'ergodicité. Rappelons que nous notons par Γ_t la variable aléatoire représentant le noyau de transition entre X_t et X_{t+1}.

4.2.1. Preuve du théorème 2.3.1

Théorème. *Soit un algorithme adaptatif sur l'espace d'états S, avec un espace de noyaux de transition $\mathcal{G}$, et ayant une distribution stationnaire $\pi(\cdot)$ identique pour toute distribution de transition $P_\gamma, \gamma \in \mathcal{G}$. Supposons que les deux conditions suivantes soient respectées :*

1. *Ergodicité uniforme simultanée : Pour tout $\epsilon > 0$, on peut trouver $T \in \mathbb{N}$ tel que*

$$\|P_\gamma^{(T)}(x,\cdot) - \pi(\cdot)\| < \epsilon, \quad \forall x \in S, \forall \gamma \in \mathcal{G}.$$

2. *Adaptation déclinante :*

$$D_t := \sup_x \|P_{\Gamma_t}(x,\cdot) - P_{\Gamma_{t-1}}(x,\cdot)\| \xrightarrow[t\to\infty]{} 0, \text{ en probabilité.}$$

Alors, le processus stochastique $X_T, t \in \mathbb{N}$, généré par cet algorithme est ergodique.

Démonstration. (Tirée de [28])

Soit $\epsilon > 0$, $T = T(\epsilon) \in \mathbb{N}$ validant la première condition et $t^* \in \mathbb{N}$ tel que

$$\mathbb{P}(D_t \geq \epsilon/T^2) \leq \epsilon/T, \forall t \geq t^*,$$

dont l'existence est garantie par la seconde condition.

Définissons l'événement $E = \bigcap_{i=t+1}^{t+T} \{D_i < \epsilon/T^2\}$. Dès que $t \geq t^*$, la probabilité de ce dernier est bornée inférieurement, ainsi :

$$\begin{aligned}\mathbb{P}(E) &= \mathbb{P}\left(\bigcap_{i=t+1}^{t+T} \{D_i < \epsilon/T^2\}\right) = 1 - \mathbb{P}\left(\bigcup_{i=t+1}^{t+T} \{D_i \geq \epsilon/T^2\}\right) \\ &\geq 1 - \sum_{i=t+1}^{t+T} \mathbb{P}(D_i \geq \epsilon/T^2) \geq 1 - \frac{T\epsilon}{T} = 1 - \epsilon. \end{aligned} \tag{4.2.1}$$

Pour la suite, supposons que E s'est effectivement produit et fixons un temps d'arrivée $K \geq T + t^*$. Alors par l'inégalité du triangle, pour tout m tel que $K - T \leq m \leq K$ et pour tout $x \in S$,

$$\begin{aligned}\|P_{\Gamma_m}(x,\cdot) - P_{\Gamma_{K-T}}(x,\cdot)\| &\leq \sum_{i=K-T}^{m} \|P_{\Gamma_{i+1}}(x,\cdot) - P_{\Gamma_i}(x,\cdot)\| \\ &\leq \sum_{i=K-T}^{m} D_{i+1} < (m - K + T)\epsilon/T^2 \leq \epsilon/T. \end{aligned} \tag{4.2.2}$$

Considérons maintenant les suites $\{X_t\}_\mathbb{N}$ et $\{\Gamma_t\}_\mathbb{N}$. Il est possible de construire une seconde chaîne $\{X'_t\}_{K-T}^K$ respectant :

1. $X'_{K-T} = X_{K-T}$,
2. $X'_t \sim P_{\Gamma_{K-T}}(X'_{t-1}, \cdot), \quad \forall t : K - T < t \leq K$,
3. $\mathbb{P}(X'_i = X_i, \forall i : K - T \leq i \leq m) \geq 1 - (m - (K - T))\epsilon/T, \quad \forall m : K - T \leq m \leq K$.

Pour le montrer, nous fixons les deux premières propriétés et montrons la troisième par induction. Pour $m = K - T$, nous avons, puisque $X'_{K-T} = X_{K-T}$,

$$\mathbb{P}(X'_{K-T} = X_{K-T}) = 1 \geq 1 - \epsilon/T,$$

tel que souhaité.

Si l'affirmation est vraie pour un certain m, alors conditionnellement à $\{X'_i = X_i, \forall i : K - T \leq i \leq m\}$, les distributions de X_{m+1} et X'_{m+1} sont respectivement $P_{\Gamma_m}(X_m, \cdot)$ et $P_{\Gamma_{K-T}}(X_m, \cdot)$ et toujours sous l'événement E, par (4.2.2) :

$$\|P_{\Gamma_m}(X_m, \cdot) - P_{\Gamma_{K-T}}(X_m, \cdot)\| < \epsilon/T.$$

Alors, par une propriété de la distance en variation totale (propriété (4.1.2) du théorème 4.1), il est possible de construire X_{m+1} et X'_{m+1} de façon à ce que $\mathbb{P}(X'_{m+1} = X_{m+1} | X'_i = X_i, \forall i : K - T \leq i \leq m) \geq 1 - \epsilon/T$, et donc

$$\begin{aligned}
&\mathbb{P}(X'_i = X_i, \forall i : K - T \leq i \leq m + 1) \\
&\quad = \mathbb{P}(X'_{m+1} = X_{m+1} | X'_i = X_i, \forall i : K - T \leq i \leq m) \mathbb{P}(X'_i = X_i, \forall i : K - T \leq i \leq m) \\
&\quad \geq (1 - \epsilon/T)(1 - (m - (K - T))\epsilon/T) \\
&\quad = 1 - \epsilon/T - (m - (K - T))\epsilon/T + (m - (K - T))\epsilon^2/T^2 \\
&\quad = 1 - (m + 1 - (K - T))\epsilon/T + (m - (K - T))\epsilon^2/T^2 \\
&\quad \geq 1 - (m + 1 - (K - T))\epsilon/T,
\end{aligned}$$

ce qui valide la propriété pour $m + 1$ et termine la preuve par induction.

Pour $m = K$, nous avons alors

$$\mathbb{P}(X'_K \neq X_K) \leq (K - (K - T))\epsilon/T = \epsilon. \tag{4.2.3}$$

D'autre part, la condition d'ergodicité uniforme simultanée garantit l'inégalité suivante :

$$\|\mathcal{L}(X'_K) - \pi(\cdot)\| = \int_S \|P^T_{\Gamma_{K-T}}(x, \cdot) - \pi(\cdot)\| f_{X_{K-T}}(x) \mathrm{d}x < \epsilon \int_S f_{X_{K-T}}(x) \mathrm{d}x = \epsilon.$$

Ainsi, toujours par la propriété (4.1.2) de la distance en variation totale, il est possible de construire la variable aléatoire Z conditionnellement à $\{X_t\}$ et $\{X'_t\}$ de façon à ce que

$$Z \sim \pi(\cdot) \text{ et } \mathbb{P}(X'_K \neq Z) < \epsilon. \tag{4.2.4}$$

Finalement, par (4.2.1), (4.2.3) et (4.2.4), pour tout $K \geq T + t^*$,

$$\begin{aligned}
\mathbb{P}(X_K \neq Z) &= \mathbb{P}(E^c, X_K \neq Z) + \mathbb{P}(E, X_K \neq X'_K, X_K \neq Z) + \mathbb{P}(E, X_K = X'_K, X'_K \neq Z) \\
&\leq \mathbb{P}(E^c) + \mathbb{P}(E, X_K \neq X'_K) + \mathbb{P}(E, X'_K \neq Z) < 3\epsilon,
\end{aligned}$$

et donc, par (4.1.1),

$$\|\mathcal{L}(X_K) - \pi(\cdot)\| \leq \mathbb{P}(X_K \neq Z) < 3\epsilon.$$

□

4.2.2. Preuve du théorème 2.3.2

Sous les mêmes conditions (2.3.6) et (2.3.7), il est possible de prouver une loi faible des grands nombres pour la chaîne adaptative $X_1, \ldots, X_n$, c'est-à-dire que pour toute fonction $g : S \to \mathbb{R}$ bornée et mesurable et pour toutes valeurs initiales $x \in S$ et $\gamma \in \mathcal{G}$,

$$\frac{\sum_{i=1}^{t} g(X_i)}{t} \xrightarrow[t\to\infty]{p} \pi(g).$$

Démonstration. (D'après [28]) Nous noterons $\mathbb{E}_{\gamma,x}[\cdot], \gamma \in \mathcal{G}, x \in S$, l'espérance basée sur une chaîne de Markov (non-adaptative) de noyau de transition P_γ et de valeur initiale x, et $\mathbb{E}[\cdot]$ l'espérance par rapport à la chaîne adaptative d'intérêt. Supposons aussi sans perte de généralité que $\pi(g) = 0$.

La loi des grands nombres pour les chaînes de Markov ergodiques nous assure que pour x et γ fixés,

$$\mathbb{E}_{\gamma,x}\left[\left|\frac{\sum_{i=1}^{t} g(X_i)}{t}\right|\right] \xrightarrow[t\to\infty]{} \pi(g) = 0.$$

Alors, par la condition d'ergodicité uniforme (2.3.6), pour tout $\epsilon > 0$, il existe $T \in \mathbb{N}$ tel que pour tout x et γ,

$$\mathbb{E}_{\gamma,x}\left[\left|\frac{\sum_{i=1}^{T} g(X_i)}{T}\right|\right] < \epsilon.$$

Comme dans la preuve du théorème 2.3.1, la seconde condition (2.3.7) nous assure l'existence de $t^* \in \mathbb{N}$ tel que pour tout $t \geq t^*$, advenant un certain événement E de probabilité supérieure à $1 - \epsilon$ (voir (4.2.1)), il est possible de construire une chaîne de Markov $X'_{t+1}, \ldots, X'_{t+T}$ de noyau P_{Γ_n} telle que

$$\mathbb{P}(X_k = X'_k,\ \forall k \in \{t+1, \ldots, t+T\}) \geq 1 - \epsilon.$$

Alors, en notant $M = \sup_S(|g(x)|)$ et F l'événement que $X_k = X'_k$ pour tout $k \in \{t+1, \ldots, t+T\}$.

$$\begin{aligned}
\mathbb{E}\left[\left|\frac{\sum_{i=t+1}^{t+T} g(X_i)}{T}\right|\right] &\leq \mathbb{E}\left[\left|\frac{\sum_{i=t+1}^{t+T} g(X_i)}{T}\right| \mathbb{I}(E \cap F)\right] + \mathbb{E}\left[\left|\frac{\sum_{i=t+1}^{t+T} g(X_i)}{T}\right| \mathbb{I}(E^c)\right] \\
&\quad + \mathbb{E}\left[\left|\frac{\sum_{i=t+1}^{t+T} g(X_i)}{T}\right| \mathbb{I}(F^c)\right] \\
&= \mathbb{E}_{\Gamma_n, X_n}\left[\left|\frac{\sum_{i=t+1}^{t+T} g(X_i)}{T}\right| \mathbb{I}(E \cap F)\right] + \mathbb{E}\left[\left|\frac{\sum_{i=t+1}^{t+T} g(X_i)}{T}\right| \mathbb{I}(E^c)\right] \\
&\quad + \mathbb{E}\left[\left|\frac{\sum_{i=t+1}^{t+T} g(X_i)}{T}\right| \mathbb{I}(F^c)\right]
\end{aligned}$$

$$\leq \mathbb{E}_{\Gamma_n, X_n}\left[\left|\frac{\sum_{i=t+1}^{t+T} g(X_i)}{T}\right|\right] + M\mathbb{P}(E^c) + M\mathbb{P}(F^c)$$
$$\leq \epsilon(1+2M). \tag{4.2.5}$$

Soit $N \in \mathbb{N}$ choisi de façon à être supérieur à MT/ϵ et à Mt^*/ϵ. Nous allons séparer $|\sum_{i=1}^N g(X_i)/N|$ en trois parties :

$$\left|\frac{1}{N}\sum_{i=1}^{N} g(X_i)\right| = \frac{1}{N}\left|\sum_{i=1}^{t^*} g(X_i) + \sum_{i=t^*+1}^{t^*+\lfloor\frac{N-t^*}{T}\rfloor T} g(X_i) + \sum_{i=t^*+\lfloor\frac{N-t^*}{T}\rfloor T+1}^{N} g(X_i)\right|. \tag{4.2.6}$$

Par la définition de N, l'espérance de la valeur absolue des premier et troisième termes est inférieure à ϵ. Quant au second terme,

$$\begin{aligned}\left|\frac{1}{N}\sum_{i=t^*+1}^{t^*+\lfloor\frac{N-t^*}{T}\rfloor T} g(X_i)\right| &\leq \left|\frac{1}{\lfloor(N-t^*)/T\rfloor}\frac{1}{T}\sum_{i=t^*+1}^{t^*+\lfloor\frac{N-t^*}{T}\rfloor T} g(X_i)\right| \\ &= \left|\frac{1}{\lfloor(N-t^*)/T\rfloor}\frac{1}{T}\sum_{j=1}^{\lfloor(N-t^*)/T\rfloor}\sum_{k=1}^{T} g(X_{t^*+(j-1)T+k})\right| \\ &= \frac{1}{\lfloor(N-t^*)/T\rfloor}\sum_{j=1}^{\lfloor(N-t^*)/T\rfloor}\left|\frac{1}{T}\sum_{k=1}^{T} g(X_{t^*+(j-1)T+k})\right|.\end{aligned}$$

Alors, par (4.2.5),

$$\begin{aligned}\mathbb{E}\left[\left|\frac{1}{N}\sum_{i=t^*+1}^{t^*+\lfloor\frac{N-t^*}{T}\rfloor T} g(X_i)\right|\right] &\leq \frac{1}{\lfloor(N-t^*)/T\rfloor}\sum_{j=1}^{\lfloor(N-t^*)/T\rfloor}\mathbb{E}\left[\left|\frac{1}{T}\sum_{k=1}^{T} g(X_{t^*+(j-1)T+k})\right|\right] \\ &\leq \frac{1}{\lfloor(N-t^*)/T\rfloor}\sum_{j=1}^{\lfloor(N-t^*)/T\rfloor}\epsilon(1+2M) \leq \epsilon(1+2M). \end{aligned}\tag{4.2.7}$$

Ainsi, en prenant l'espérance de chacun des termes de (4.2.6), nous avons

$$\mathbb{E}\left[\left|\frac{1}{N}\sum_{i=1}^{N} g(X_i)\right|\right] \leq 2\epsilon + \epsilon(1+2M) = \epsilon(3+2M). \tag{4.2.8}$$

L'inégalité de Markov nous permet alors de conclure :

$$\mathbb{P}\left(\left|\frac{1}{N}\sum_{i=1}^{N} g(X_i)\right| \geq \epsilon^{1/2}\right) \leq \frac{\mathbb{E}\left[\left|\frac{1}{N}\sum_{i=1}^{N} g(X_i)\right|\right]}{\epsilon^{1/2}} \leq \epsilon^{1/2}(3+2M).$$

□

Remarque 4.2.1. *La première condition d'ergodicité (2.3.6) peut s'avérer difficile à évaluer directement et des critères plus accessibles existent. Entre autres, il y aura ergodicité uniforme si l'espace $S \times \mathcal{G}$ est compact selon une certaine topologie dans laquelle la fonction $(x,\gamma) \mapsto \|P_\gamma(x,\cdot) - \pi(\cdot)\|$ est continue pour tout t. De même, cette continuité sera vérifiée si les*

distributions de proposition $Q_\gamma(x,\cdot)$ possèdent toutes une densité $q_\gamma(x,.)$ dont l'ensemble est uniformément borné et tel que $(x,\gamma) \mapsto q_\gamma(x,x')$ est continue, pour tout $x' \in S$. Ce sera le cas entre autres des propositions normales, du moment que $S \times \mathcal{G}$ est compact. On pourra se référer à [28] pour la démonstration de ces énoncés.

4.2.3. Preuve du théorème 2.3.3

On déduit de ces derniers résultats que l'algorithme AM présenté précédemment est effectivement ergodique. Pour faciliter la démonstration, nous supposerons que la densité cible $\pi(\cdot)$ est continue et strictement positive sur son support S compact. Ces conditions sont peu contraignantes en pratique, mais notons tout de même que celles-ci peuvent être affaiblies par une argumentation plus élaborée, voir [16]. La démonstration utilise de nombreux résultats intermédiaires sur l'algèbre linéaire et les densités normales, dont la validité est démontrée en annexe A.

Démonstration. (D'après [28].)

Tout d'abord, notons que l'ajout de ϵI_d à une matrice de covariance quelconque C nous assure que cette dernière est définie positive, donc qu'il existe une constante c_1 telle que pour toute matrice C_t, la matrice $C_t - c_1 I_d$ est définie semi-positive (voir proposition A.1.4). D'autre part, la compacité de S nous assure une borne sur la covariance de la chaîne. Par le corollaire A.1.1, ceci implique que l'on peut trouver des constantes c_1 et c_2 telles que

$$c_1 I_d \leq C \leq c_2 I_d, \tag{4.2.9}$$

dans le sens où $C_t - c_1 I_d$ et $c_2 I_d - C_t$ sont définies semi-positives. Ces deux bornes assurent la compacité de l'espace $\mathcal{G}$ (voir la section A.1.3) et donc de $S \times \mathcal{G}$. L'utilisation de distributions instrumentales normales termine la validation de la première condition (2.3.6), d'après les propriétés que nous venons d'énoncer (remarque 4.2.1).

Pour prouver la seconde condition (2.3.7), nous utiliserons les contraintes ajoutées sur la densité $\pi(\cdot)$. Celles-ci nous permettent de borner cette dernière supérieurement et inférieurement par des constantes strictement positives. Par la compacité de S et (4.2.9), nous avons également de telles bornes uniformes sur toutes les densités de proposition $f_{\Gamma_t}(x,y)$ (voir la proposition A.2.1). Ceci garantit l'existence des valeurs suivantes :

$$M = \max\left(\sup_{x\in S} \pi(x), \sup_{x,y\in S,\gamma\in\mathcal{G}} f_\gamma(x,y)\right) \text{ et} \tag{4.2.10}$$

$$\epsilon = \min\left(\inf_{x\in S} \pi(x), \inf_{x,y\in S,\gamma\in\mathcal{G}} f_\gamma(x,y)\right). \tag{4.2.11}$$

Nous avons alors la série d'inégalités suivante

$$\sup_x \| P_{\Gamma_{t+1}}(x,\cdot) - P_{\Gamma_t}(x,\cdot) \|$$

$$
\begin{aligned}
&= \sup_x \sup_A \left| \int_A \left(P_{\Gamma_{t+1}}(x,y) - P_{\Gamma_t}(x,y) \right) \mathrm{d}y \right| \\
&= \sup_x \sup_A \Bigg| \int_A f_{\Gamma_{t+1}}(x,y)\alpha_{t+1}(x,y)\mathrm{d}y - \int_A f_{\Gamma_t}(x,y)\alpha_t(x,y)\mathrm{d}y \\
&\quad + \mathbb{I}(x \in A) \left(\int_S f_{\Gamma_{t+1}}(x,y)(1-\alpha_{t+1}(x,y))\mathrm{d}y - \int_S f_{\Gamma_t}(x,y)(1-\alpha_t(x,y))\mathrm{d}y \right) \Bigg| \\
&\leq \sup_x \sup_A \int_A \left| f_{\Gamma_{t+1}}(x,y)\alpha_{t+1}(x,y) - f_{\Gamma_t}(x,y)\alpha_t(x,y) \right| \mathrm{d}y \\
&\quad + \mathbb{I}(x \in A) \int_S \left| f_{\Gamma_{t+1}}(x,y)(1-\alpha_{t+1}(x,y)) - f_{\Gamma_t}(x,y)(1-\alpha_t(x,y)) \right| \mathrm{d}y \\
&\leq \sup_x \int_S \left| f_{\Gamma_{t+1}}(x,y)\alpha_{t+1}(x,y) - f_{\Gamma_t}(x,y)\alpha_t(x,y) \right| \mathrm{d}y \\
&\quad + \int_S \left| f_{\Gamma_{t+1}}(x,y)(1-\alpha_{t+1}(x,y)) - f_{\Gamma_t}(x,y)(1-\alpha_t(x,y)) \right| \mathrm{d}y \\
&\leq \sup_x 2 \int_S \left| f_{\Gamma_{t+1}}(x,y) - f_{\Gamma_t}(x,y) \right| \mathrm{d}y \\
&\leq 2 \left(\sup_{x,y} \left| f_{\Gamma_{t+1}}(x,y) - f_{\Gamma_t}(x,y) \right| \right) \left(\int_S \mathrm{d}y \right). \qquad (4.2.12)
\end{aligned}
$$

Nous voulons donc montrer la convergence vers 0 de cette dernière expression. L'espace S étant compact, l'intégrale est finie et il suffit donc de démontrer que les différences successives entre les densités de proposition tendent vers 0 uniformément en x et y quand $t \to \infty$, ce que nous faisons maintenant.

Rappelons que dans cet algorithme, $f_{\Gamma_t}(x,y)$ est une densité normale de moyenne x et de matrice de covariance C_t. La compacité de l'espace S nous assure l'existence d'une borne sur toutes les composantes de toutes les observations x_t. Alors, par les formules récursives (2.3.3) et (2.3.4), nous avons

$$
\begin{aligned}
\overline{x}_t \overline{x}_t^T &= \frac{t^2}{(t+1)^2} \overline{x}_{t-1} \overline{x}_{t-1}^T + \frac{t}{(t+1)^2} \left(x_t \overline{x}_{t-1}^T + \overline{x}_{t-1} x_t^T \right) + \frac{1}{(t+1)^2} x_t x_t^T, \\
C_{t+1} - C_t &= -\frac{1}{t} C_t + \frac{s_d}{t} \left(t \overline{x}_{t-1} \overline{x}_{t-1}^T - (t+1) \overline{x}_t \overline{x}_t^T + x_t x_t^T + \epsilon I_d \right) \\
&= -\frac{1}{t} C_t + s_d \left(\frac{1}{t+1} \overline{x}_{t-1} \overline{x}_{t-1}^T - \frac{1}{t+1} \left(x_t \overline{x}_{t-1}^T + \overline{x}_{t-1} x_t^T \right) + \frac{1}{t+1} x_t x_t^T + \frac{\epsilon}{t} I_d) \right) \\
&\xrightarrow[t \to \infty]{} \mathbf{0}, \qquad (4.2.13)
\end{aligned}
$$

dans le sens où chaque coefficient de $C_{t+1} - C_t$ tend vers 0. Par opérations sur les limites, ceci implique que $\det C_t - \det C_{t+1} \to 0$ et $C_{t+1}^{-1} - C_t^{-1} \to 0$ quand $t \to \infty$. Notons également que l'inégalité (4.2.9) implique que pour tout t, $\det(C_t) \geq \det(c_1 I) = c_1^d$ (voir section A.1.3). Par la proposition A.2.2, ces propriétés sont suffisantes pour conclure que

$$
\sup_{x,y} |f_{\Gamma_{t+1}}(x,y) - f_{\Gamma_t}(x,y)| \xrightarrow[t \to \infty]{} 0, \qquad (4.2.14)
$$

ce qui termine la preuve de la seconde condition d'ergodicité. □

4.3. Ergodicité de l'algorithme RAPT

Nous présentons une preuve d'ergodicité détaillée, puisque l'ergodicité de notre propre algorithme s'appuiera sur des arguments très semblables. Rappelons d'abord la forme de la densité de proposition pour l'algorithme RAPT :

$$f_{\Gamma_t}(x,y) = (1-\beta)\sum_{i=1}^{K}\mathbb{I}_{(X_t\in S_i)}\sum_{j=1}^{K}\lambda_{ij}^{(t)}q_j^{(t)}(x,y) + \beta q_S^{(t)}(x,y), \tag{4.3.1}$$

où les $q_j^{(t)}(x,y)$ sont de densité normale centrée en x et de covariance respective $C_j^{(t)}$ pour $j \in \{1,\cdots,K,S\}$ et où $\Gamma_t = (\lambda_{11}^{(t)},\ldots,\lambda_{KK}^{(t)},C_1^{(t)},\ldots,C_K^{(t)},C_S^{(t)})$ représente l'ensemble des paramètres adaptatifs à un instant t donné.

Théorème 4.3.1. *Pour un espace échantillonnal S compact et une densité cible $\pi(\cdot)$ continue et strictement positive sur S, l'algorithme RAPT de [6] tel que présenté dans la section 3.2.2 respecte bien les conditions suffisantes d'ergodicité ((2.3.6) et (2.3.7)) des algorithmes adaptatifs.*

Démonstration. (Généralisation basée sur [6] et [23])

Notons d'abord des bornes M et ϵ analogues à celles définies dans la preuve précédente ((4.2.10) et (4.2.11)) existent et sont strictement positives. Ceci est assuré par la compacité de S, la continuité et la positivité sur S des densités $\pi(\cdot)$ et q_j, ainsi que par la construction des matrices adaptatives C_j, $j \in \{1,\ldots,K,S\}$.

4.3.1. Ergodicité uniforme

Pour tous $x \in S, \gamma \in \mathcal{G}$ et $B \subset S$ mesurable fixés, définissons

$$R_{x,\gamma}(B) := \left\{ y \in B : \frac{\pi(y)f_\gamma(y,x)}{\pi(x)f_\gamma(x,y)} < 1 \right\}.$$

Alors, la probabilité qu'une chaîne à l'état x à un instant donné se trouve dans B au temps suivant satisfait

$$\begin{aligned}
P_\gamma(x,B) &\geq \int_B f_\gamma(x,y)\min\left(\frac{\pi(y)f_\gamma(y,x)}{\pi(x)f_\gamma(x,y)},1\right)\mathrm{d}y \\
&= \int_{R_{x,\gamma}(B)} \frac{\pi(y)f_\gamma(y,x)}{\pi(x)}\mathrm{d}y + \int_{B\setminus R_{x,\gamma}(B)} f_\gamma(x,y)\frac{\pi(y)}{\pi(y)}\mathrm{d}y \\
&\geq \frac{\epsilon}{M}\int_{R_{x,\gamma}(B)}\pi(y)\mathrm{d}y + \frac{\epsilon}{M}\int_{B\setminus R_{x,\gamma}(B)}\pi(y)\mathrm{d}y \\
&= \frac{\epsilon}{M}\pi(B).
\end{aligned}$$

La mesure $\nu(B) := \pi(B)\epsilon/M$ est non-triviale sur S et ceci nous assure, par l'argumentation qui suit, que

$$\|P_\gamma^{(t)}(x,\cdot) - \pi(\cdot)\| \leq (1-\epsilon/M)^t, \tag{4.3.2}$$

pour tous x, γ, avec bien sûr $0 < (1-\epsilon/M) < 1$, ce qui garantit l'ergodicité uniforme simultanée, tel que souhaité.

Pour montrer (4.3.2), posons $\rho = 1-\epsilon/M$ et construisons les chaînes $\{U_t\}$ et $\{V_t\}$, avec $U_0 = x$ et $V_0 \sim \pi(\cdot)$, et les règles de transition suivantes à chaque étape $t \in \mathbb{N}$:

1. Avec probabilité $1-\rho$: $U_t \sim \nu(\cdot)/\rho$ et $V_t = U_t$;

2. Avec probabilité ρ : $U_t \sim \left(P_\gamma(U_{t-1},\cdot) - (1-\rho)\nu(\cdot)/\rho\right)/\rho$. Si la première étape a déjà été effectuée au moins une fois, on pose $V_t = U_t$. Sinon, V_t est indépendante de U_t et de distribution $\left(P_\gamma(V_{t-1},\cdot) - (1-\rho)\nu(\cdot)/\rho\right)/\rho$.

On vérifie facilement que $U_t \sim P_\gamma^{(t)}(x,\cdot)$ et $V_t \sim \pi(\cdot)$ pour tout t. En effet, nous avons

$$\mathbb{P}(U_t = \mathrm{d}y | U_{t-1}) = (1-\rho)\frac{\nu(\mathrm{d}y)}{\rho} + \frac{\rho}{\rho}\left(P_\gamma(U_{t-1},\cdot) - (1-\rho)\frac{\nu(\mathrm{d}y)}{\rho}\right)/\rho = P_\gamma(U_{t-1},\cdot),$$

et une relation analogue pour $\{V_t\}$. Ainsi, $\{U_t\}$ et $\{V_t\}$ ont comme distribution de transition $P_\gamma(\cdot,\cdot)$, ce qui implique que $\{U_t\}$ est de distribution $P_\gamma^{(t)}(x,\cdot)$ et que $\{V_t\}$ est stationnaire. De plus, avec T représentant le premier indice de temps où $U_t = V_t$,

$$P(U_t \neq V_t) \leq \mathbb{P}(T > t) \leq \rho^t.$$

Ainsi, par (4.1.1),

$$\|P_\gamma^{(t)}(x,\cdot) - \pi(\cdot)\| \leq P(U_t \neq V_t) \leq \rho^t.$$

4.3.2. Adaptation déclinante

Soit $A \in S$ et supposons sans perte de généralité que $x \in S_1$. On aura alors :

$$\begin{aligned} P_{\Gamma_t}(x,A) = &\int_{A\cap S_1} f_{\Gamma_t}(x,y)\alpha_t^*(x,y)\mathrm{d}y \\ &+ \cdots + \int_{A\cap S_K} f_{\Gamma_t}(x,y)\alpha_t^*(x,y)\mathrm{d}y \\ &+ \mathbb{I}(x\in A)\int_{S_1} f_{\Gamma_t}(x,y)(1-\alpha_t^*(x,y))\mathrm{d}y \\ &+ \cdots + \mathbb{I}(x\in A)\int_{S_K} f_{\Gamma_t}(x,y)(1-\alpha_t^*(x,y))\mathrm{d}y, \end{aligned} \tag{4.3.3}$$

où α_t^* est défini comme en (3.2.6). Notons chacun des termes dans (4.3.3) par $I_{i,t}(x,A)$, $i = 1, \ldots, 2K$. Nous avons alors

$$\begin{aligned} \sup_x \sup_A |P_{\Gamma_{t+1}}(x,A) - P_{\Gamma_t}(x,A)| &\leq \sup_x \sup_A \sum_{i=1}^{2K} |I_{i,t+1}(x,A) - I_{i,t}(x,A)| \\ &\leq \sup_x \sum_{i=1}^{2K} |I_{i,t+1}(x,S) - I_{i,t}(x,S)|. \end{aligned} \tag{4.3.4}$$

Pour prouver la seconde condition, nous voudrons donc montrer que chacun des termes à droite tend vers 0, en leur appliquant un traitement semblable à (4.2.12). Nous le ferons de façon détaillée uniquement pour le cas $i = 2$, la preuve pour les autres termes étant très semblable. Pour ce terme, nous avons l'inégalité suivante :

$$\begin{aligned} &|I_{2,t+1}(x,S) - I_{2,t}(x,S)| \\ &= \left| \int_{S_2} f_{\Gamma_{t+1}}(x,y)\alpha_{t+1}^*(x,y)\mathrm{d}y - \int_{S_2} f_{\Gamma_t}(x,y)\alpha_t^*(x,y)\mathrm{d}y \right| \\ &\leq \int_{S_2} |f_{\Gamma_{t+1}}(x,y)\alpha_{t+1}^*(x,y) - f_{\Gamma_t}(x,y)\alpha_t^*(x,y)|\mathrm{d}y \\ &= \int_{S_2} |f_{\Gamma_{t+1}}(x,y)\alpha_{t+1}^*(x,y) - f_{\Gamma_{t+1}}(x,y)\alpha_t^*(x,y) \\ &\quad + f_{\Gamma_{t+1}}(x,y)\alpha_t^*(x,y) - f_{\Gamma_t}(x,y)\alpha_t^*(x,y)|\mathrm{d}y \\ &\leq \int_{S_2} f_{\Gamma_{t+1}}(x,y)|\alpha_{t+1}^*(x,y) - \alpha_t^*(x,y)|\mathrm{d}y \\ &\quad + \int_{S_2} \alpha_t^*(x,y)|f_{\Gamma_{t+1}}(x,y) - f_{\Gamma_t}(x,y)|\mathrm{d}y. \end{aligned}$$

Traitons maintenant l'argument de la première intégrale uniquement :

$$\begin{aligned} &f_{\Gamma_{t+1}}(x,y)|\alpha_{t+1}^*(x,y) - \alpha_t^*(x,y)|\mathrm{d}y \\ &\leq M\frac{\pi(y)}{\pi(x)}\left|\frac{f_{\Gamma_{t+1}}(y,x)}{f_{\Gamma_{t+1}}(x,y)} - \frac{f_{\Gamma_t}(y,x)}{f_{\Gamma_t}(x,y)}\right| \\ &\leq \frac{M^2}{\epsilon}\left|\frac{f_{\Gamma_{t+1}}(y,x)f_{\Gamma_t}(x,y) - f_{\Gamma_t}(y,x)f_{\Gamma_{t+1}}(x,y)}{f_{\Gamma_{t+1}}(x,y)f_{\Gamma_t}(x,y)}\right| \\ &\leq \frac{M^2}{\epsilon^3}|f_{\Gamma_{t+1}}(y,x)f_{\Gamma_t}(x,y) - f_{\Gamma_t}(y,x)f_{\Gamma_t}(x,y) \\ &\quad + f_{\Gamma_t}(y,x)f_{\Gamma_t}(x,y) - f_{\Gamma_t}(y,x)f_{\Gamma_{t+1}}(x,y)| \\ &\leq \frac{M^2}{\epsilon^3}\left(f_{\Gamma_t}(x,y)|f_{\Gamma_{t+1}}(y,x) - f_{\Gamma_t}(y,x)| + f_{\Gamma_t}(y,x)|f_{\Gamma_{t+1}}(x,y) - f_{\Gamma_t}(x,y)|\right) \\ &\leq \frac{M^3}{\epsilon^3}\left(|f_{\Gamma_{t+1}}(y,x) - f_{\Gamma_t}(y,x)| + |f_{\Gamma_{t+1}}(x,y) - f_{\Gamma_t}(x,y)|\right). \end{aligned}$$

Ceci donne finalement :

$$\begin{aligned}
&\sup_x |I_{2,t+1}(x,A) - I_{2,t}(x,A)| \\
&\quad \leq \left(\frac{M^3}{\epsilon^3} + 1\right) \sup_x \int_{S_2} \left(|f_{\Gamma_{t+1}}(y,x) - f_{\Gamma_t}(y,x)| + |f_{\Gamma_{t+1}}(x,y) - f_{\Gamma_t}(x,y)| \right) \mathrm{d}y \\
&\quad \leq 2\left(\frac{M^3}{\epsilon^3} + 1\right) \left(\sup_{x,y} |f_{\Gamma_{t+1}}(x,y) - f_{\Gamma_t}(x,y)| \right) \left(\int_S \mathrm{d}y \right).
\end{aligned}$$

Pour terminer, il faut donc vérifier la convergence uniforme vers 0 de $|f_{\Gamma_{t+1}}(x,y) - f_{\Gamma_t}(x,y)|$. Par construction de f (voir (4.3.1)) et sous l'hypothèse que $x \in S_1$, nous avons

$$\begin{aligned}
&|f_{\Gamma_{t+1}}(x,y) - f_{\Gamma_t}(x,y)| \\
&= \left| (1-\beta) \sum_{j=1}^{K} \left(\lambda_{1j}^{(t+1)} q_j^{(t+1)}(x,y) - \lambda_{1j}^{(t)} q_j^{(t)}(x,y) \right) + \beta (q_S^{(t+1)}(x,y) - q_S^{(t)}(x,y)) \right| \\
&\leq (1-\beta) \sum_{j=1}^{K} \left(q_j^{(t)}(x,y) \left| \lambda_{1j}^{(t+1)} - \lambda_{1j}^{(t)} \right| + \lambda_{1j}^{(t)} \left| q_j^{(t+1)}(x,y) - q_j^{(t)}(x,y) \right| \right) \\
&\quad + \beta |q_S^{(t+1)}(x,y) - q_S^{(t)}(x,y)|.
\end{aligned}$$

Rappelons que tous les paramètres et densités sont bornés. De plus, l'adaptation des matrices de covariance étant la même que dans l'algorithme AM, par la preuve du théorème précédent, nous avons que pour tout j approprié, $|q_j^{(t+1)}(x,y) - q_j^{(t)}(x,y)| \to 0$ quand $t \to \infty$, uniformément pour tous $x,y \in S$. Ainsi, il ne reste qu'à montrer que $|\lambda_{1j}^{(t+1)} - \lambda_{1j}^{(t)}| \to 0$, pour tout j (revoir (3.2.4) pour la définition de ces paramètres). Dans le cas où la région S_1 n'est jamais explorée, $\lambda_{1j}^{(t+1)} = \lambda_{1j}^{(t)} = 1/K$ pour tout t et nous avons terminé. De même, si S_1 n'est visitée qu'un nombre fini de fois, alors à partir d'un t suffisamment grand nous aurons toujours $|\lambda_{11}^{(t+1)} - \lambda_{11}^{(t)}| = 0$. Sinon, nous avons la forme suivante, où nous avons posé $j = 1$, sans perte de généralité :

$$|\lambda_{11}^{(t+1)} - \lambda_{11}^{(t)}| = \left| \frac{d_{11}^{(t+1)}}{\sum_{j=1}^K d_{1j}^{(t+1)}} - \frac{d_{11}^{(t)}}{\sum_{j=1}^K d_{1j}^{(t)}} \right| = \frac{|d_{11}^{(t+1)} \sum_{j=2}^K d_{1j}^{(t)} - d_{11}^{(t)} \sum_{j=2}^K d_{1j}^{(t+1)}|}{\sum_{j=1}^K d_{1j}^{(t+1)} \sum_{j=1}^K d_{1j}^{(t)}}. \tag{4.3.5}$$

Cette expression sera nulle pour tous les temps t tels que $x_t \notin S_1$, ainsi que pour les temps t où y_{t+1} a été sélectionné par la distribution $Q_S^{(t)}$. Pour les autres temps t, la forme dépendra de la distribution $Q_j^{(t)}, j \in \{1, \ldots, K\}$ choisie. Nous nous contenterons du cas où $x_t \in S_1$ et $Y_{t+1} \sim Q_1^{(t)}(\cdot)$, les autres cas étant très semblables. Dans ce cas, nous avons que $d_{1j}^{(t+1)} = d_{1j}^{(t)}$ pour tout $j \neq 1$. Par (3.2.2) et (3.2.3), l'expression (4.3.5) devient alors

$$\frac{\sum_{j=2}^K d_{1j}^{(t)}}{\sum_{j=1}^K d_{1j}^{(t+1)} \sum_{j=1}^K d_{1j}^{(t)}} \left| d_{11}^{(t+1)} - d_{11}^{(t)} \right|$$

$$= \frac{\sum_{j=2}^{K} d_{1j}^{(t)}}{\sum_{j=1}^{K} d_{1j}^{(t+1)} \sum_{j=1}^{K} d_{1j}^{(t)}} \left| \frac{|W_{11}^{(t)}| d_{11}^{(t)} + \|X_{t+1} - X_t\|_2^2}{|W_{11}^{(t)}| + 1} - d_{11}^{(t)} \right|$$
$$= \frac{\sum_{j=2}^{K} d_{1j}^{(t)}}{\sum_{j=1}^{K} d_{1j}^{(t+1)} \sum_{j=1}^{K} d_{1j}^{(t)}} \frac{\left| \|X_{t+1} - X_t\|_2^2 - d_{11}^{(t)} \right|}{|W_{11}^{(t)}| + 1},$$

où $\|\cdot\|_2$ est la norme euclidienne. Notons que la compacité de S nous assure que toutes les quantités au numérateur sont bornées. Pour que le tout tende vers zéro, on voudra aussi qu'au moins l'une des distances moyennes cumulatives $d_{1j}^{(t)}$ ne tende pas vers 0, ce qui est équivalent à demander que la chaîne ne dégénère pas vers un seul point. Or, cela sera toujours le cas dans le contexte étudié ici. En effet, la construction des matrices $C_j^{(t)}$ assure qu'elles sont en quelque sorte bornées inférieurement (voir propriété (4.2.9)), ce qui implique qu'à tout moment le candidat proposé Y_{t+1} sera différent de X_t avec probabilité 1. De même, les bornes ((4.2.10) et (4.2.11)) sur $\pi(\cdot)$ et $f(\cdot,\cdot)$ nous assurent d'une borne inférieure strictement positive sur les seuils d'acceptation α^* spécifiés en (3.2.6). Ainsi, avec probabilité 1, les candidats proposés seront acceptés une infinité de fois, empêchant la chaîne de dégénérer.

Finalement, il faudra que $W_{11}^{(t)} \to \infty$ quand $t \to \infty$, autrement dit que pour une infinité de temps t, $Y_{t+1} \sim Q_1^{(t)}$ (rappelons que nous sommes déjà dans le cas où S_1 est visité un nombre infini de fois). Or, si cette condition n'est pas remplie, à partir d'un certain point, cette forme particulière de $|\lambda_{11}^{(t+1)} - \lambda_{11}^{(t)}|$ ne se produira plus. Puisqu'il est certain qu'au moins l'une des densités sera utilisée un nombre infini de fois, nous pouvons supposer sans perte de généralité qu'il s'agit de Q_1. Ainsi, nous avons bien

$$|\lambda_{11}^{(t+1)} - \lambda_{11}^{(t)}| \xrightarrow[t\to\infty]{} 0,$$

ce qui entraîne

$$|f_{\Gamma_{t+1}}(x,y) - f_{\Gamma_t}(x,y)| \xrightarrow[t\to\infty]{} 0,$$

uniformément pour tous $x,y \in S$, et finalement

$$\sup_x |I_{2,t+1}(x,S) - I_{2,t}(x,S)| \xrightarrow[t\to\infty]{} 0.$$

Par une argumentation similaire, il en va de même pour chacun des termes

$$\sup_x |I_{i,t+1}(x,S) - I_{i,t}(x,S)|, \ i \in \{1,\ldots,2K\}$$

de (4.3.4), ce qui termine la preuve de la seconde condition. □

4.4. Ergodicité de l'algorithme OPRA

Nous démontrons dans cette section l'ergodicité de notre algorithme dans le cas où l'espace échantillonnal S est divisé en 2 régions et nous donnons des indications pour la généralisation de la preuve à n'importe quel nombre fini K de régions.

4.4.1. Considérations théoriques

Afin d'assurer le respect des conditions suffisantes d'ergodicité (2.3.6) et (2.3.7), il nous faudra ajouter une petite étape assurant que les moyennes empiriques calculées dans chaque région ne se rapprochent pas arbitrairement l'une de l'autre, même si en pratique ceci ne risque pas de survenir. Après le calcul de $\hat{\mu}_1(t+1)$ et $\hat{\mu}_2(t+1)$, si

$$\|\hat{\mu}_1^{(t+1)} - \hat{\mu}_2^{(t+1)}\|_2 < \delta,$$

pour un certain $\delta > 0$ fixé à l'avance, on perturbera légèrement $\hat{\mu}_2^{(t+1)}$ en lui ajoutant le vecteur

$$\left(\frac{\delta}{\|\hat{\mu}_1^{(t+1)} - \hat{\mu}_2^{(t+1)}\|_2} - 1\right)(\hat{\mu}_2^{(t+1)} - \hat{\mu}_1^{(t+1)}). \tag{4.4.1}$$

Ceci a pour effet d'éloigner $\hat{\mu}_2^{(t+1)}$ de $\hat{\mu}_1^{(t+1)}$ dans la direction du vecteur joignant les deux moyennes, jusqu'à ce que la distance entre les deux soit exactement δ. En choisissant δ suffisamment petit, cette modification aura un impact négligeable sur les performances de l'algorithme en pratique et simplifiera grandement la démonstration qui va suivre. Nous pouvons maintenant énoncer le théorème.

4.4.2. Preuve d'ergodicité

Théorème 4.4.1. *Pour un espace échantillonnal S compact et une densité cible $\pi(\cdot)$ continue et strictement positive sur S, l'algorithme OPRA, tel que décrit à la section 3.4 et muni de l'étape supplémentaire (4.4.1), respecte les conditions suffisantes d'ergodicité des algorithmes adaptatifs.*

Démonstration. Tout d'abord, la preuve de la condition d'ergodicité uniforme simultanée (2.3.6) est identique à celle du RAPT, étant donné que les nouveaux paramètres adaptatifs ne changent en rien l'existence de bornes sur la densité de proposition et la densité cible ou leur positivité, seuls éléments nécessaires à cette portion de la preuve.

La preuve de la seconde condition (2.3.7) est aussi semblable, mais avec la complication supplémentaire que les régions d'intégration dans (4.3.3) et (4.3.4) varient maintenant avec t. Ainsi, par exemple, en posant $g_{\Gamma_t}(x,y) := f_{\Gamma_t}(x,y)\alpha_t^*(x,y)$, avec f et α^* définis comme en (4.3.1) et (3.2.6), le terme $|I_{2,t+1}(x,A) - I_{2,t}(x,A)|$ de (4.3.4) devient

$$\begin{aligned}
&\left| \int_{S_2^{(t+1)}} g_{\Gamma_{t+1}}(x,y)\mathrm{d}y - \int_{S_2^{(t)}} g_{\Gamma_t}(x,y)\mathrm{d}y \right| \\
&\qquad = \left| \int_{S_2^{(t)} \cap S_2^{(t+1)}} g_{\Gamma_{t+1}}(x,y)\mathrm{d}y - \int_{S_2^{(t)} \cap S_2^{(t+1)}} g_{\Gamma_t}(x,y)\mathrm{d}y \right. \\
&\qquad\quad \left. + \int_{S_1^{(t)} \cap S_2^{(t+1)}} g_{\Gamma_{t+1}}(x,y)\mathrm{d}y - \int_{S_1^{(t+1)} \cap S_2^{(t)}} g_{\Gamma_t}(x,y)\mathrm{d}y \right|.
\end{aligned} \tag{4.4.2}$$

Pour se ramener à la preuve précédente, il faut donc montrer que les deux derniers termes tendent vers zéro quand $t \to \infty$. La fonction g étant uniformément bornée, il suffit de montrer que la mesure du sous-ensemble de S changeant de régions entre les temps t et $t+1$ tend vers 0. L'argumentation qui suit suppose que $d \geq 2$, nous traiterons à la fin le cas où $d = 1$.

À un temps donné t, pour que deux hyperplans séparateurs successifs soient exactement parallèles ou confondus, il faudrait que X_{t+1} soit généré exactement sur la droite joignant $\hat{\mu_1}^{(t)}$ et $\hat{\mu_2}^{(t)}$, ce qui se produira avec probabilité 0. Ainsi, à chaque étape, les hyperplans successifs s'intersectent et leur intersection est alors un hyperplan de dimension $d-2$. L'angle aigu entre les plans successifs (en fait l'angle entre leurs vecteurs normaux a_t et a_{t+1}) est donné par

$$\theta_t := \arccos\left(\frac{a_t^T a_{t+1}}{\|a_t\|_2 \|a_{t+1}\|_2}\right),$$

avec a_t défini comme en (3.4.1). Alors, avec $i(t)$ représentant l'indice de la région (1 ou 2) d'appartenance de x_{t+1},

$$\begin{aligned} a_{t+1} &= \hat{\mu}_1^{(t+1)} - \hat{\mu}_2^{(t+1)} \\ &= \hat{\mu}_1^{(t)} - \hat{\mu}_2^{(t)} + \frac{(-1)^{i(t)-1}(x_{t+1} - \hat{\mu}_{i(t)}^{(t)})}{|W_{i(t)}^{(t+1)}|} \\ &= a_t + \frac{(-1)^{i(t)-1}(x_{t+1} - \hat{\mu}_{i(t)}^{(t)})}{|W_{i(t)}^{(t+1)}|}. \end{aligned} \tag{4.4.3}$$

Nous désignerons par c_t le second terme de (4.4.3). Par la compacité de S, chacune des composantes de c_t est bornée au numérateur. De plus, $|W_{i(t)}^{(t+1)}| \to \infty$ quand $t \to \infty$, car même dans l'éventualité où l'une des régions, disons S_1, n'est visitée qu'un nombre fini de fois, alors pour t assez grand, $i(t)$ vaudra toujours 2 et on aura bien $|W_2^{(t+1)}| \to \infty$. Ainsi, $c_t \to 0_d$ composante par composante. Alors, en utilisant (4.4.3) et l'inégalité du triangle au dénominateur,

$$\begin{aligned} 1 &\geq \frac{a_t^T a_{t+1}}{\|a_t\|_2 \|a_{t+1}\|_2} \\ &= \frac{\|a_t\|_2^2 + a_t^T c_t}{\|a_t\|_2 \|a_t + c_t\|_2} \\ &\geq \frac{\|a_t\|_2^2 + a_t^T c_t}{\|a_t\|_2^2 + \|a_t\|_2 \|c_t\|_2} \\ &= 1 - \frac{\|a_t\|_2 \|c_t\|_2 - a_t^T c_t}{\|a_t\|_2^2 + \|a_t\|_2 \|c_t\|_2} \xrightarrow[t\to\infty]{} 1, \end{aligned} \tag{4.4.4}$$

puisque la compacité de S et les spécificités de l'algorithme, notamment la perturbation (4.4.1), assurent l'existence de constantes δ et M_0 telles que $0 < \delta \leq \|a_t\|_2 \leq M_0 < \infty$ pour tout t. Finalement, la continuité de la fonction arccos implique que $\theta_t \to 0$ quand $t \to \infty$.

Ainsi, l'angle entre les hyperplans successifs tend vers zéro. Le volume délimité par ceux-ci et les limites de l'ensemble compact S tendra donc lui aussi vers 0, du moment que l'intersection des hyperplans successifs ne s'éloigne pas arbitrairement de S. Or, on peut vérifier qu'effectivement cela ne peut se produire. En effet, à une étape t donnée, le point z^* le plus près de l'origine étant contenu dans l'intersection des deux hyperplans successifs peut être déterminé en minimisant la fonction suivante par les multiplicateurs de Lagrange :

$$h(z) = \|z\|_2^2 + 2\lambda_1(a_t^T z + b_t) + 2\lambda_2(a_{t+1}^T z + b_{t+1}). \tag{4.4.5}$$

Le point recherché doit donc satisfaire le système suivant :

$$\begin{aligned} z^* + \lambda_1 a_t + \lambda_2 a_{t+1} &= 0_d, \\ a_t^T z^* - b_t &= 0, \\ a_{t+1}^T z^* - b_{t+1} &= 0. \end{aligned}$$

Après quelques calculs, on détermine que

$$z^* = \frac{\left(\|a_{t+1}\|_2^2 b_t - a_t^T a_{t+1} b_{t+1}\right) a_t + \left(\|a_t\|_2^2 b_{t+1} - a_t^T a_{t+1} b_t\right) a_{t+1}}{\|a_t\|_2^2 \|a_{t+1}\|_2^2 - (a_t^T a_{t+1})^2}. \tag{4.4.6}$$

Par une décomposition semblable à celle de l'équation (4.4.4), on a que le dénominateur est borné inférieurement par une constante positive. De plus, tous les paramètres de chacun des plans sont bornés, étant fonctions uniquement de moyennes empiriques sur un espace compact S. Ainsi, chaque composante de z^* est bornée, et il en va de même de $\|z^*\|_2$, ce qui implique que la distance entre l'intersection de deux hyperplans adaptatifs consécutifs et S est également bornée, tel que souhaité. Notons que ceci est également valable pour la version avec pondération de l'hyperplan par les covariances, telle que décrite à la section 3.4.4. En effet, seul le calcul du paramètre b_t est modifié, et celui-ci reste borné puisque $b_t = a_t^T \left((1-k_t)\hat{\mu}_1^{(t)} + k_t \hat{\mu}_2^{(t)}\right)$, avec $k_t \in (0,1)$.

Finalement donc, chacune des régions d'intégration d'intérêt est contenue dans un sous-ensemble de $\mathbb{R}^d$ formé par le produit cartésien d'un hypercube de dimension $d-2$ et d'un secteur de disque dont l'angle tend vers 0. La mesure d'un tel ensemble tend bien sûr vers 0. La fonction $g_{\Gamma_t}(x,y)$ étant uniformément bornée, disons par M_1, nous avons alors, en repartant de (4.4.2),

$$\begin{aligned} &\sup_x |I_{2,t+1}(x,S) - I_{2,t}(x,S)| \\ &\qquad \leq \sup_x \left| \int_{S_2^{(t)} \cap S_2^{(t+1)}} g_{\Gamma_{t+1}}(x,y)\mathrm{d}y - \int_{S_2^{(t)} \cap S_2^{(t+1)}} g_{\Gamma_t}(x,y)\mathrm{d}y \right| \\ &\qquad\quad + \left| \int_{S_1^{(t)} \cap S_2^{(t+1)}} g_{\Gamma_{t+1}}(x,y)\mathrm{d}y \right| + \left| \int_{S_1^{(t+1)} \cap S_2^{(t)}} g_{\Gamma_t}(x,y)\mathrm{d}y \right| \end{aligned}$$

$$\leq \sup_x \int_{S_2^{(t)} \cap S_2^{(t+1)}} \left| g_{\Gamma_{t+1}}(x,y) - g_{\Gamma_t}(x,y) \right| \mathrm{d}y$$
$$+ M_1 \int_{S_1^{(t)} \cap S_2^{(t+1)}} \mathrm{d}y + M_1 \int_{S_1^{(t+1)} \cap S_2^{(t)}} \mathrm{d}y \xrightarrow[t\to\infty]{} 0,$$

la convergence du premier terme ayant été montrée dans la preuve d'ergodicité de l'algorithme RAPT et celle des autres termes découlant de l'argumentation précédente. La conclusion se poursuit alors de la même façon que pour la preuve précédente et la seconde condition d'ergodicité est donc bien respectée. □

4.4.3. Remarques

Si $d = 1$, alors les hyperplans sont en fait des points dans $\mathbb{R}$ donnés au temps t par $d_t = (\hat{\mu}_1^{(t)} + \hat{\mu}_2^{(t)})/2$. Alors, par un développement semblable à (4.4.3), nous avons que

$$d_{t+1} = d_t + c_t^*,$$

avec $c_t^* \to 0$ quand $t \to \infty$, et donc,

$$|d_{t+1} - d_t| \xrightarrow[t\to\infty]{} 0.$$

Ainsi, toute intégrale sur le segment entre d_t et d_{t+1} d'une fonction bornée tendra vers zéro. Nous présentons cet argument par souci de complétude, bien qu'en pratique ce genre d'algorithme ne soit pas particulièrement utile pour des cibles unidimensionnelles.

Pour étendre la preuve aux cas à plus de deux régions, notons que nous pouvons réécrire l'expression (4.4.2) de façon plus générale ainsi :

$$\left| \int_{S_2^{(t+1)}} g_{\Gamma_{t+1}}(x,y)\mathrm{d}y - \int_{S_2^{(t)}} g_{\Gamma_t}(x,y)\mathrm{d}y \right|$$
$$= \left| \int_{S_2^{(t)} \cap S_2^{(t+1)}} g_{\Gamma_{t+1}}(x,y)\mathrm{d}y - \int_{S_2^{(t)} \cap S_2^{(t+1)}} g_{\Gamma_t}(x,y)\mathrm{d}y \right.$$
$$\left. + \int_{\left(S_2^{(t)}\right)^c \cap S_2^{(t+1)}} g_{\Gamma_{t+1}}(x,y)\mathrm{d}y - \int_{\left(S_2^{(t+1)}\right)^c \cap S_2^{(t)}} g_{\Gamma_t}(x,y)\mathrm{d}y \right|,$$

et la borner supérieurement par

$$\left| \int_{S_2^{(t)} \cap S_2^{(t+1)}} g_{\Gamma_{t+1}}(x,y)\mathrm{d}y - \int_{S_2^{(t)} \cap S_2^{(t+1)}} g_{\Gamma_t}(x,y)\mathrm{d}y \right|$$
$$+ \sum_{i\neq 2, 1\leq i\leq K} \left| \int_{S_i^{(t)} \cap S_2^{(t+1)}} g_{\Gamma_{t+1}}(x,y)\mathrm{d}y \right| + \sum_{i\neq 2, 1\leq i\leq K} \left| \int_{S_i^{(t+1)} \cap S_2^{(t)}} g_{\Gamma_t}(x,y)\mathrm{d}y \right|.$$

On peut alors montrer que chacun des termes de la deuxième ligne tend vers 0, par une argumentation analogue à celle de la preuve qui précède, étant donné que chaque région d'intégration dépend du mouvement d'un seul hyperplan séparateur. Par exemple, l'aire de

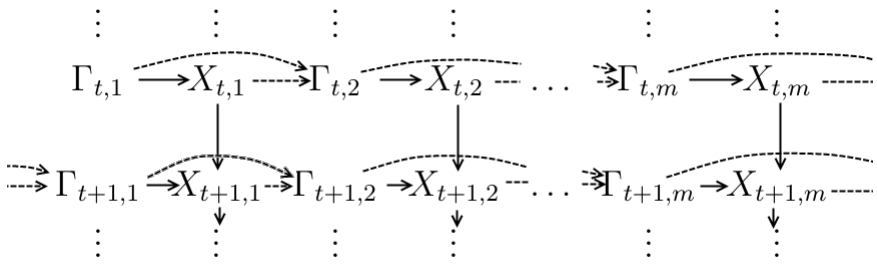

FIGURE 4.1. Représentation du processus d'adaptation interchaînes. Les calculs se font ligne par ligne de gauche à droite. Les flèches simples représentent l'action du noyau de transition de chaque chaîne et les flèches pointillées l'action de la fonction g qui met à jour les paramètres adaptatifs.

la région $S_i^{(t+1)} \cap S_2^{(t)}$ ne dépend que de l'évolution entre les temps t et $t+1$ de l'hyperplan séparant $\hat{\mu}_i$ et $\hat{\mu}_2$, et ainsi de suite.

4.5. Ergodicité des algorithmes parallèles

Un processus d'adaptation interchaînes ayant été ajouté aux algorithmes RAPT, RAPTOR et OPRA dans plusieurs de nos expériences de simulation, nous montrons ici qu'un tel ajout préserve bien l'ergodicité de ces algorithmes. Les processus d'adaptation et de génération d'un algorithme avec adaptation interchaîne sont formalisés dans le théorème qui suit.

Théorème 4.5.1. *Pour un espace échantillonnal S compact et une densité cible $\pi(\cdot)$ continue et strictement positive sur S, considérons le processus MCMC à m chaînes parallèles suivant :*

1. *$X_{0,1}, \ldots, X_{0,m}$ sont choisis au hasard dans S,*
2. *Au temps $t+1$, dans la chaîne i :*
   (a) *Le candidat $Y_{t+1,i}$ provient de la distribution $Q_{\Gamma_{t+1,i}}(X_{t,i}, \cdot)$, où Q est la distribution instrumentale des algorithmes RAPT et OPRA, dont la densité est donnée en (4.3.1).*
   (b) *On met à jour l'ensemble des paramètres adaptatifs à partir de l'information obtenue jusqu'à maintenant dans toutes les chaînes. De façon récursive, nous avons donc $\Gamma_{t+1,i} = g(\Gamma_{t+1,i-1}, X_{t+1,i-1})$ pour $i = 2, \ldots, m$, et $\Gamma_{t+1,1} = g(\Gamma_{t,m}, X_{t,m})$ pour $i = 1$, où $g : \mathcal{G} \times S \to \mathcal{G}$ est la fonction mettant à jour de façon récursive les paramètres, donc basée sur (2.3.3), (2.3.4), (3.2.2), (3.2.4), (3.4.1) et (3.4.2). Le schéma de la figure 4.1 résume la situation.*

Alors, chacune des chaînes $X_{0,i}, X_{1,i}, \ldots$, avec $i \in \{1, \ldots, m\}$ respecte les conditions suffisantes d'ergodicité (2.3.6) et (2.3.7) des algorithmes adaptatifs.

Démonstration. Comme nous l'avons vu dans la preuve pour l'algorithme RAPT, la condition d'ergodicité uniforme simultanée ne dépend que de l'existence de bornes inférieures et supérieures sur les densités instrumentale et cible. Le processus à l'étude ici ne changeant rien à l'ensemble des densités instrumentales possibles, cette argumentation tient toujours pour chacune des chaînes $\{X_{t,i}\}_{t\in\mathbb{N}}$.

D'autre part, dans les démonstrations précédentes, nous avons montré que

$$\sup_x \|P_{\Gamma_{t+1}}(x,\cdot) - P_{\Gamma_t}(x,\cdot)\| \xrightarrow[t\to\infty]{p} 0,$$

pour tous les algorithmes adaptatifs étudiés. Ce résultat ne dépendant pas des valeurs particulières générées par la chaîne, nous avons de façon équivalente, dans le contexte qui nous intéresse ici que

$$\sup_x \|P_{\Gamma_{t+1,i}}(x,\cdot) - P_{\Gamma_{t+1,i-1}}(x,\cdot)\| \xrightarrow[t\to\infty]{p} 0,$$

pour $i = 2,\ldots,m$ et

$$\sup_x \|P_{\Gamma_{t+1,1}}(x,\cdot) - P_{\Gamma_{t,m}}(x,\cdot)\| \xrightarrow[t\to\infty]{p} 0.$$

Or, par l'inégalité du triangle, nous avons, pour chaque $i \in \{1,\ldots,m\}$:

$$\begin{aligned}\sup_x \|P_{\Gamma_{t+1,i}}(x,\cdot) - P_{\Gamma_{t,i}}(x,\cdot)\| \leq & \sum_{k=i+1}^{m} \sup_x \|P_{\Gamma_{t,k}}(x,\cdot) - P_{\Gamma_{t,k-1}}(x,\cdot)\| \\ & + \sup_x \|P_{\Gamma_{t+1,1}}(x,\cdot) - P_{\Gamma_{t,m}}(x,\cdot)\| \\ & + \sum_{k=2}^{i} \sup_x \|P_{\Gamma_{t,k}}(x,\cdot) - P_{\Gamma_{t,k-1}}(x,\cdot)\| \xrightarrow[t\to\infty]{p} 0,\end{aligned}$$

ce qui valide la condition d'adaptation déclinante pour chacune des chaînes. □

Chapitre 5

MESURES DE PERFORMANCES

Nous présentons ici plusieurs méthodes permettant de déterminer la qualité d'un échantillon MCMC et de comparer des échantillons obtenus par diverses techniques. Certains de ces critères vérifient la convergence du processus. D'autres tentent de déterminer dans quelle mesure la chaîne explore rapidement l'espace échantillonnal et si elle reproduit convenablement la distribution cible. La combinaison de ces méthodes peut nous aider à choisir l'algorithme le plus approprié ainsi que ses paramètres pour une classe de problèmes donnée. Ainsi, plusieurs de ces méthodes ont été employées pour comparer la qualité de notre algorithme par rapport aux autres algorithmes avec adaptation régionale.

5.1. Méthodes graphiques

Un simple graphique des valeurs successives de l'échantillon peut nous en apprendre beaucoup sur le comportement de la chaîne. En principe, un algorithme performant devrait couvrir adéquatement l'ensemble des valeurs possibles et traverser l'espace d'intérêt dans un nombre relativement petit d'itérations. De plus, la présence de plateaux sur la même valeur ou l'absence de grands sauts dans le parcours suggère que la distribution candidate n'est pas adaptée à la situation.

Il est aussi possible de tracer un graphe de certaines statistiques empiriques (moyenne, variance, quantiles) cumulatives en fonction du nombre d'itérations prises en compte. On est alors en mesure de vérifier visuellement s'il semble y avoir convergence vers certaines valeurs et, dans un contexte d'expérimentation, de comparer celles-ci aux valeurs attendues.

5.2. Taux de rejet

Le taux de rejet d'une chaîne est une autre statistique simple à calculer et souvent révélatrice. Il n'y a rien d'étonnant à ce que la majorité des valeurs soient rejetées, même en petite dimension, mais un taux trop élevé ou trop bas indique souvent un échantillon inapproprié, probablement causé par une distribution candidate inadaptée ou un mauvais choix d'algorithme. Pour les algorithmes adaptatifs, une telle situation suggère que l'adaptation ne s'est

pas complétée. Il faudrait alors envisager une augmentation du nombre d'itérations, une modification de la matrice initiale C_0, ou encore une réévaluation du temps de pré-adaptation t_0. Sous certaines conditions, il a été montré (voir [10]) qu'un algorithme de Metropolis-Hastings optimal accepte environ 23.4% des candidats proposés. Pour des cibles bimodales ou non-convexes comme celles que nous utiliserons plus loin, ce taux d'acceptation théorique ne s'applique pas, mais il serait considéré inhabituel d'observer des taux d'acceptation très éloignés de cette valeur. À titre d'exemple, toutes les chaînes que nous avons simulées et qui ont bien convergé selon d'autres critères avaient des taux d'acceptation dans l'intervalle 0.234 ± 0.06.

5.3. Erreur quadratique moyenne

On peut évaluer à quel point un échantillon a bien estimé une statistique d'intérêt θ de valeur connue en mesurant la distance quadratique entre la statistique estimée sur l'échantillon $\hat{\theta}$ et sa vraie valeur θ. En combinant l'information de plusieurs réplications indépendantes de chaînes menant aux estimés $\hat{\theta}_1, \ldots, \hat{\theta}_N$, il est possible de calculer une erreur quadratique moyenne empirique de la façon suivante :

$$E\hat{Q}M(\theta) = \frac{1}{N}\sum_{i=1}^{N}\|\hat{\theta}_i - \theta\|_2^2,$$

que l'on souhaitera bien sûr aussi petite que possible.

5.4. Variation quadratique moyenne

La variation quadratique moyenne, ou distance de saut moyenne, ou AQV (pour *Average quadratic variation*), est une mesure d'efficacité très simple. Pour une chaîne de $T+1$ éléments indicés de 0 à T, elle est définie ainsi :

$$\text{AQV} = \sum_{t=1}^{T}\frac{\|X_t - X_{t-1}\|_2^2}{T}. \tag{5.4.1}$$

Ainsi, cette quantité mesure l'ampleur moyenne des déplacements entre les valeurs consécutives de la chaîne, que l'on souhaite aussi grands que possible. Elle peut aussi être calculée de façon récursive à chaque étape de l'algorithme dans le but d'étudier son évolution.

5.5. Autocorrélation

On définit ainsi l'autocorrélation avec décalage $k \geq 0$ entre deux suites de variables réelles x_t et y_t, $t = 1, \ldots, T$ (où potentiellement $\{x_t\} = \{y_t\}$) ainsi :

$$\rho_k = \frac{T\sum_{t=1}^{T-k}(x_t - \overline{x})(y_{t+k} - \overline{y})}{(T-k)\left(\sum_{t=1}^{T}(x_t - \overline{x})^2\right)^{1/2}\left(\sum_{t=1}^{T}(y_t - \overline{y})^2\right)^{1/2}}. \tag{5.5.1}$$

On voit clairement le parallèle avec le coefficient de corrélation empirique standard appliqué aux vecteurs $\{x_t \mid t = 1, \ldots, n-k\}$ et $\{y_t \mid t = k, \ldots, n\}$. Dans le cas des MCMC, on s'intéressera surtout à l'autocorrélation entre une suite de coordonnées de notre échantillon et elle-même. Il va de soi que l'autocorrélation sera relativement forte pour des petites valeurs de k. Par contre, on souhaitera que ρ_k diminue à mesure que le décalage k augmente, jusqu'à devenir négligeable, car autrement cela signifierait une forte corrélation à long terme entre les éléments de la chaîne et donc une exploration lente de l'espace échantillonnal. Il faudrait alors envisager une augmentation du nombre d'itérations, ou encore revoir le processus. Un graphe de l'autocorrélation en fonction de k devrait montrer une courbe décroissante et on préférera les algorithmes présentant la décroissance la plus rapide.

5.6. Mesure du taux de couverture

L'utilisation de cette méthode est restreinte aux distributions cibles pour lesquelles il est facile de déterminer si un point fait partie d'une région de confiance d'un niveau donné. Il ne sera donc pas possible de l'appliquer à toutes les situations, mais elle demeure tout de même une mesure très utile permettant de mesurer et de comparer la performance des algorithmes dans des situations relativement variées, en les appliquant à des distributions connues ou à des transformations de ces dernières. À titre d'exemple, considérons une cible de loi normale. Pour une telle distribution, une valeur donnée x sera dans la région de confiance de niveau $1-\alpha$ si et seulement si

$$(x-\mu)^T \Sigma^{-1} (x-\mu) < \chi^2_{(d,1-\alpha)}, \tag{5.6.1}$$

où $\mu \in \mathbb{R}^d$ et $\Sigma \in \mathbb{R}^{d \times d}$ sont respectivement le vecteur moyen et la covariance de la distribution, et $\chi^2_{(d,k)}$ est le k-ième quantile d'une distribution χ^2 avec d degrés de liberté. Pour estimer le taux de couverture, on procède alors ainsi :

1. Pour un algorithme donné, on génère un certain nombre k de chaînes indépendantes.
2. On choisit un niveau de confiance $1-\alpha$, et on calcule pour chacune des k chaînes la proportion de valeurs se trouvant dans la région de confiance théorique de niveau $1-\alpha$.
3. On calcule finalement la moyenne et la variance empiriques des k taux de couvertures.

Si l'échantillon représente bien la distribution cible, la moyenne devrait être très proche du niveau de confiance théorique $1-\alpha$ et la variance relativement petite. En général, on voudra un k relativement grand pour avoir des estimés précis. Pour comparer des algorithmes différents, on effectuera simplement la même procédure pour chacun. De la même façon, on pourrait tenter d'optimiser grossièrement la valeur d'un certain paramètre d'un algorithme en le faisant varier progressivement. En général, pour obtenir un diagnostic plus complet, et ce, sans grand impact sur le temps de calcul puisque les échantillons sont déjà générés, on mesurera les taux de couverture à plusieurs niveaux de confiance différents, en prenant soin

d'inclure des valeurs près de 1. Ceci permettra de vérifier si les extrêmes de la distribution sont bien représentés par l'échantillon.

Finalement, il est intéressant de pouvoir effectuer ces mesures sur certaines distributions irrégulières, par exemple de densité $f \circ g(x)$, où f est la densité normale et g une transformation non-linéaire du vecteur x. Il est aisé de calculer le taux de couverture théorique dans cette nouvelle situation en remplaçant simplement x par $g(x)$ en (5.6.1).

Lors des simulations, nous souhaiterons calculer le taux de couverture lorsque la distribution cible de densité $\pi(\cdot)$ est un mélange de distributions normales. Dans ce cas, il n'y a pas de relation simple comparable à (5.6.1). Par contre, comme une telle distribution est facile à simuler par des méthodes conventionnelles, il est possible d'effectuer une estimation par simulation des taux de couverture théoriques, par exemple par la méthode suivante.

1. Générer un grand nombre N d'observations X_i^* i.i.d. de la distribution cible ;
2. Ordonner les valeurs $\pi(X_i^*)$ pour obtenir $\pi(X^*)_{(1)} \leq \pi(X^*)_{(2)} \leq \cdots \leq \pi(X^*)_{(N)}$;
3. Estimer la borne $\hat{K}_\alpha = \pi(X^*)_{(\lfloor \alpha N \rfloor)}$;

Une observation x serait alors considérée à l'intérieur de la région de confiance estimée de niveau $1 - \alpha$ si $\pi(x) \geq \hat{K}_\alpha$. On pourrait aussi au besoin répéter cette procédure plusieurs fois pour obtenir en plus une idée de la volatilité de l'estimateur $\hat{K}_\alpha$.

5.7. Autres critères de convergence

Il existe de nombreux autres critères permettant de vérifier la stabilisation d'une chaîne MCMC lorsque nous avons peu d'information sur la distribution cible. Comme nos simulations dans le cadre de ce projet avaient pour cibles des distributions aux propriétés bien connues, soit des mélanges de distributions normales, ces autres critères sont d'un intérêt limité pour la présente recherche et ne sont donc que cités brièvement ici.

Le critère R de Brooks-Gelman-Rubin, élaboré dans [5], est basé sur la génération de nombreuse chaînes parallèles indépendantes. L'idée est de mesurer la variation intra-chaîne et interchaînes d'une certaine quantité d'intérêt θ fonction des échantillons, puis de construire deux estimateurs asymptotiquement sans biais de $\mathrm{Var}(\theta)$ basés sur ces mesures, l'un sous-estimant la vraie valeur et l'autre la surestimant. Ainsi, une condition nécessaire pour la convergence du processus serait que le rapport entre ces deux quantités se rapproche de 1.

Le diagnostic de Geweke [11] consiste à comparer la moyenne de la première portion de la chaîne (habituellement les premiers 10%) avec celle de la dernière portion (habituellement les derniers 50%), au moyen d'un test standard de différence de moyennes.

Le diagnostic de Raftery-Lewis (voir [12] et [25]) estime à partir d'une chaîne pilote le nombre d'itérations à effectuer et le nombre de valeurs initiales de la chaîne à rejeter pour qu'un certain quantile q d'une composante de la densité cible soit correctement estimé à

l'intérieur d'un seuil de tolérance r avec une certaine probabilité s, où p, r et s sont choisis par l'utilisateur.

Le diagnostic d'Heidelberg-Welch [20] vérifie la stationnarité de l'échantillon obtenu par une série de tests éliminant progressivement les valeurs initiales de la chaîne.

Dans [9], il est question de certaines méthodes permettant d'estimer l'erreur standard associée à l'estimation d'un paramètre donné par un échantillon MCMC, ainsi que d'un critère d'arrêt basé sur cette mesure.

Chapitre 6

RÉSULTATS NUMÉRIQUES

Dans ce chapitre, nous étudierons le comportement de notre algorithme en pratique et le comparerons à ses compétiteurs dans diverses situations à l'aide de simulations. Nous verrons d'abord des illustrations graphiques du processus d'adaptation de la partition de notre algorithme dans le cas de cibles en deux dimensions. Nous évaluerons ensuite comment les trois types d'algorithmes à adaptation régionale se comparent sur de nombreux exemples de distributions cibles en faible dimension. Ces premiers tests permettront d'évaluer la rapidité et la flexibilité des processus d'adaptation respectifs. Nous comparerons ensuite la qualité des algorithmes à l'aide de chaînes de grande taille pour des cibles en dimension modérément élevée, c'est-à-dire des exemples plus typiques de ce que ces algorithmes seront appelés à faire en pratique, pour ensuite dresser un résumé des principaux résultats observés. Avant toute chose, nous détaillerons les critères de performance utilisés ainsi que la nature des distributions cibles et la terminologie utilisée pour les spécifier.

6.1. Remarques préliminaires

6.1.1. Critères utilisés

Nous avons utilisé comme principaux critères de comparaisons entre les algorithmes :

1. la distance quadratique entre la moyenne échantillonnale et la vraie moyenne de la distribution cible EQM_e,
2. la variation quadratique moyenne de la chaîne, AQV ;
3. la différence entre les taux de couverture empiriques et les niveaux de confiance théoriques de 50%, 90%, 95% et 99% ;

Dans les cas où les seuils théoriques correspondant aux niveaux de confiance ne pouvaient être déterminés analytiquement, ils ont été estimés de la façon décrite dans la section 5.6 sur un échantillon simulé de taille 10^6. Il est à noter que l'utilisation unique des deux premiers critères, quoique fréquente dans la littérature, peut mener à des conclusions trompeuses, comme nous le verrons plus loin. Par exemple, surtout dans le cas d'une cible symétrique, il

est possible qu'un échantillon produise un très bon estimé de la moyenne tout en couvrant mal les queues de la distribution. Au contraire, les taux de couverture empiriques calculés à de nombreux niveaux évaluent assez directement à quel point un échantillon représente bien la distribution cible et seront donc habituellement notre mesure principale de qualité en cas de désaccord entre les critères. Nous avons bien sûr aussi tenu compte du temps requis pour compléter la génération de l'échantillon dans nos analyses. Pour assurer la précision de nos conclusions, chaque scénario de simulation a été répliqué un grand nombre fois (qui sera précisé pour chaque cas) et les moyennes des critères ainsi que leurs écarts-types ont été considérés.

En plus de ces critères, nous avons dans plusieurs cas rapporté d'autres mesures qui, bien qu'elles ne puissent permettre directement de départager la qualité des algorithmes, peuvent donner certaines informations complémentaires sur la stratégie qu'un algorithme donné utilise pour apprendre la distribution cible. Celles-ci ont été essentiellement considérées dans le cas de cibles bimodales. Nous les énumérons ici :

- Le taux d'acceptation (voir chapitre précédent), T_a.
- Le taux de concordance entre les régions artificielles attribuées aux valeurs de l'échantillon par l'algorithme et les véritables régions d'appartenance de ces points, évaluées en déterminant lequel des modes possède la plus forte densité à ces endroits. Formellement, on le calcule ainsi :

  $$T_c := \frac{1}{n+1} \sum_{t=0}^{n} \left(\mathbb{I}\left(x_t \in \left(S_{01} \cap S_1^{(t)} \right) \cup \left(S_{02} \cap S_2^{(t)} \right) \right) \right).$$

  Un taux proche de 1 indiquerait que les régions définies par l'algorithme sont assez semblables à une partition naturelle des deux modes.
- Le taux de changement de région, soit la fréquence à laquelle on observe un changement de région entre deux valeurs consécutives d'une chaîne, et qui permet d'évaluer le flot entre les deux modes estimés. Ce taux peut être calculé tant sur les régions estimées par l'algorithme que sur les régions réelles, mais sera seulement utilisé dans ce premier cas et sera noté TC_e.
- La différence absolue entre les fréquences observées dans les deux régions. Pour les régions réelles S_{01} et S_{02}, nous aurons donc

  $$D_r := \frac{1}{n+1} \left| \sum_{t=0}^{n} \left(\mathbb{I}(x_t \in S_{01}) - \mathbb{I}(x_t \in S_{02}) \right) \right|,$$

  alors que pour les régions estimées progressives $S_1^{(t)}$ et $S_2^{(t)}$ nous aurons

  $$D_e := \frac{1}{n+1} \left| \sum_{t=0}^{n} \left(\mathbb{I}\left(x_t \in S_1^{(t)} \right) - \mathbb{I}\left(x_t \in S_2^{(t)} \right) \right) \right|.$$

Dans les cas où les modes sont bien séparés, on souhaitera que D_r soit aussi proche que possible de $|p-(1-p)| = |2p-1|$, p étant le poids associé au premier mode, tel que défini dans la section qui suit.

6.1.2. Spécification des distributions

Les distributions cibles sont toutes des instances de la famille de distributions suivante, paramétrée par les vecteurs μ_1 et $\mu_2 \in \mathbb{R}^d$, les matrices définies positives Σ_1 et $\Sigma_2 \in \mathbb{R}^{d\times d}$ et les constantes $p \in [0,1]$ et $\psi \geq 0$. La forme générale de la densité, définie pour toute valeur $x \in \mathbb{R}^d$ est la suivante :

$$pf(\phi_\psi(x)|\mu_1,\Sigma_1) + (1-p)f(\phi_\psi(x)|\mu_2,\Sigma_2),$$

où $f(x|\mu,\Sigma)$ est la fonction de densité normale de vecteur moyen μ et de covariance Σ, et où $\phi_\psi(x_2) = x_2 + \psi(x_1^2 - 100)$ et $\phi_\psi(x_j) = x_j$ pour les autres composantes. Ainsi, le paramètre ψ permet d'introduire une non-linéarité dans l'argument de l'exponentielle, ce qui mène à des distributions que l'on pourrait qualifier de tordues, avec des régions de confiance non-convexes. Ce type de distribution a été utilisé par exemple par [15] pour des tests comparatifs. La torsion sera plus prononcée à mesure que le facteur ψ augmente. Comme cas particuliers, on retrouve bien sûr les mélanges de lois normales (avec $\psi = 0$) et la loi normale multivariée standard (avec $\psi = 0$ et $p = 1$). En somme, cette famille possède beaucoup de flexibilité et permet d'évaluer les algorithmes dans de nombreuses situations.

6.1.3. Algorithmes utilisés

Les algorithmes comparés dans tous les tests sont les suivants :

- RAPT (voir la section 3.2.2) ;
- RAPTOR (voir la section 3.3.2) ;
- OPRA$_0$: Notre algorithme, version de base avec l'hyperplan à mi-distance euclidienne entre les moyennes empiriques (voir la section 3.4.2) ;
- OPRA : Notre algorithme, version avec pondération de la position de l'hyperplan par les covariances empiriques (voir la section 3.4.4).

Certains détails d'implémentation et une portion du code utilisé figurent à l'annexe C. La variante avec ré-évaluation des régions après la pré-adaptation (voir la section 3.4.3), que nous nommerons OPRA$_1$, a aussi été explorée, mais s'est rapidement révélée assez systématiquement inférieure aux autres versions.

6.1.4. Paramètres de départ

Finalement, rappelons les divers paramètres initiaux devant être choisis pour une simulation donnée. Tel que mentionné plus tôt, nous nous limiterons aux versions des algorithmes à $K = 2$ régions :

– Les 6 paramètres de la distribution. On supposera qu'en l'absence d'avis contraire, $\psi = 0$ et $p = 1$. De plus, μ_2 et Σ_2 ne seront pas mentionnés lorsque non nécessaires ;
– N, le nombre de réplications de chaque scénario ;
– n, le nombre total d'itérations du processus ;
– t_0, le nombre total d'itérations avant que ne débute l'adaptation ($t_0 < n$) ;
– k, le nombre de chaînes parallèles utilisées. Lorsque $k > 1$, les quantités n et t_0 sont divisées également entre toutes les chaînes.
– $X_0^{(j)}$, la valeur initiale de la chaîne j, pour $j = 1, \ldots, k$;
– $C_1^{(0)}, C_2^{(0)}$ et $C_S^{(0)}$, les matrices de covariance initiales ;
– a_0, b_0, les paramètres de l'hyperplan initial d'équation $a_0^T x = b_0$, séparant l'espace S en $S_1^{(0)}$ et $S_2^{(0)}$;
– $\mu_1^{(0)}$ et $\mu_2^{(0)}$, moyennes initiales pour l'algorithme RAPTOR ;
– Les poids initiaux $\lambda_{11}^{(0)}, \lambda_{12}^{(0)}, \lambda_{21}^{(0)}$ et $\lambda_{22}^{(0)}$, initialisés à 1/2 par défaut ;
– β, le poids de la composante globale, valant 0.3 par défaut.

6.2. Illustration de l'adaptation

Nous débutons par des illustrations visuelles du comportement adaptatif de notre algorithme sur des cibles en deux dimensions. Dans tous les cas, nous avons utilisé la version de base OPRA_0 avec $k = 2$ chaînes parallèles initialisées de part et d'autre de la délimitation initiale et $C_1^{(0)} = C_2^{(0)} = I_2$ et $C_S^{(0)} = 25I_2$. Les résultats graphiques sont présentés dans les figures 6.1 à 6.10 de la façon suivante : on affiche un échantillon i.i.d. de 10^4 valeurs tirées de la distribution cible en tons de gris afin d'illustrer la forme de celle-ci. La délimitation et les valeurs initiales de la chaîne sont en orangé, alors que le plan et les moyennes estimées $\hat{\mu}_1^{(n)}$ et $\hat{\mu}_2^{(n)}$ finaux sont en vert. Les ellipses bleues correspondent aux régions de confiance de niveau 95% estimées pour chacun des deux modes et globalement d'après les moyennes empiriques et les matrices de covariance C_1, C_2 et C_S finales. Pour les cibles bimodales, nous avons rapporté le taux de concordance T_c et la différence de fréquences réelles D_r observée.

Dans tous les cas, nous avons pris d'abord un très faible nombre d'itérations ($n = 200$) afin d'évaluer la rapidité d'adaptation du processus, puis le même exemple avec un nombre d'itérations plus modéré ($n = 2000$) permettant de visualiser les paramètres adaptatifs une fois stabilisés. Dans toutes les situations étudiées, nous pouvons observer que l'adaptation de l'hyperplan séparateur s'effectue rapidement, même lorsque la séparation initiale est loin d'être optimale. Évidemment, ce processus deviendra plus difficile et plus lent à mesure que la dimension augmente. Nous terminons cette portion avec quelques observations plus particulières sur chacun des exemples.

> Figures 6.1 et 6.2 : Nous voyons que malgré un plan initial complètement erroné, l'algorithme parvient à rétablir rapidement la situation. À moyen terme, la séparation

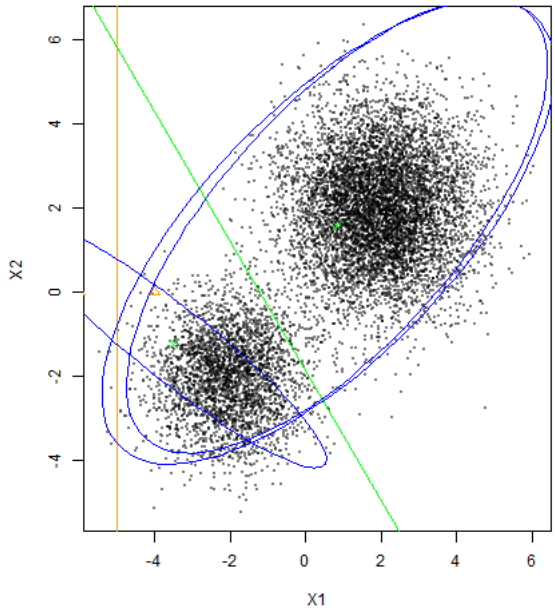

FIGURE 6.1. Illustration de l'adaptation, modes bien séparés, $n = 200$. Paramètres de la cible : $\mu_1 = (-2,-2), \mu_2 = (2,2), \Sigma_1 = I_2, \Sigma_2 = 1.5I_2, p = 0.3$. $t_0 = 100, T_c = 0.73, D_r = 0.356$.

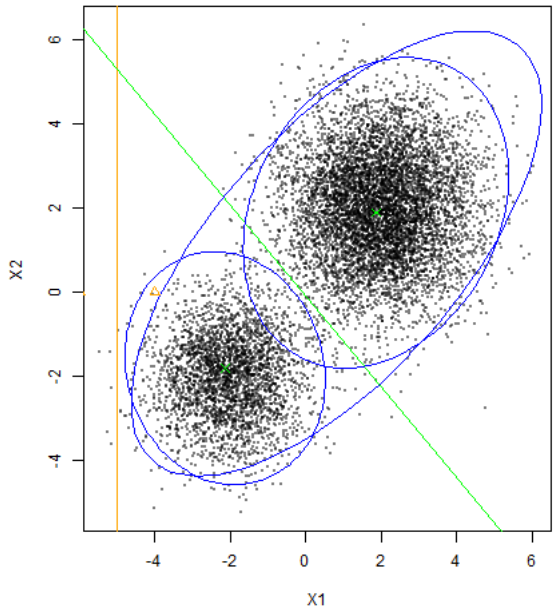

FIGURE 6.2. Illustration de l'adaptation, modes bien séparés, $n = 2000$. Paramètres de la cible : $\mu_1 = (-2,-2), \mu_2 = (2,2), \Sigma_1 = I_2, \Sigma_2 = 1.5I_2, p = 0.3$. $t_0 = 100, T_c = 0.9665, D_r = 0.4286$.

est à toutes fins pratiques optimale, tel que corroboré par le taux élevé de concordance, comme l'on pouvait s'y attendre dans une situation où les modes sont bien séparés.

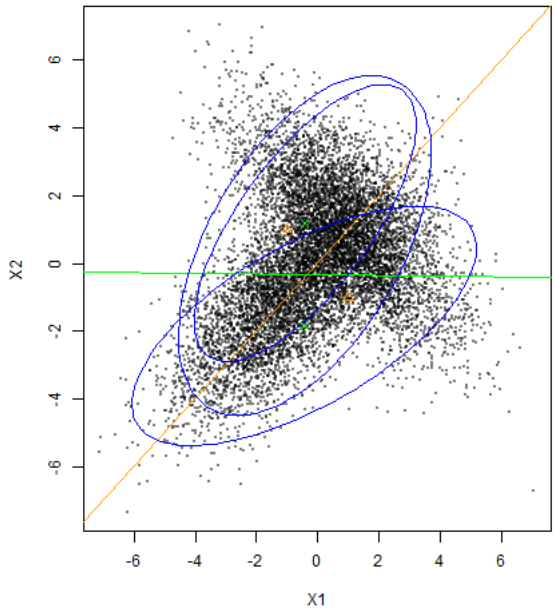

FIGURE 6.3. Illustration de l'adaptation, covariances non sphériques, $n = 200$. Paramètres de la cible : $\mu_1 = (-1,-1), \mu_2 = (1,1), \Sigma_1 = (3{,}2;{,}2{,}3), \Sigma_2 = (3,-2:-2{,}3), p = 0.5$. $t_0 = 100, T_c = 0.71, D_r = 0.20792$.

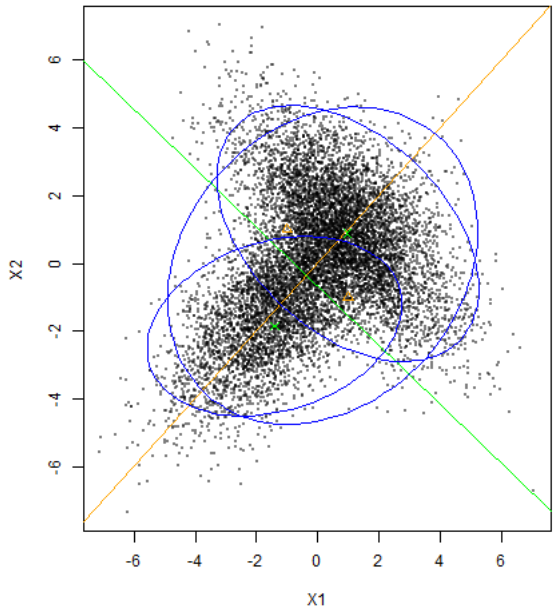

FIGURE 6.4. Illustration de l'adaptation, covariances non sphériques, $n = 2000$. Paramètres de la cible : $\mu_1 = (-1,-1), \mu_2 = (1,1), \Sigma_1 = (3{,}2;{,}2{,}3), \Sigma_2 = (3,-2:-2{,}3), p = 0.5$. $t_0 = 100, T_c = 0.876, D_r = 0.167832$.

Figures 6.3 et 6.4 : Nous remarquons que les covariances empiriques s'adaptent convenablement aux formes de chaque région. Même dans un cas comme celui-ci où les

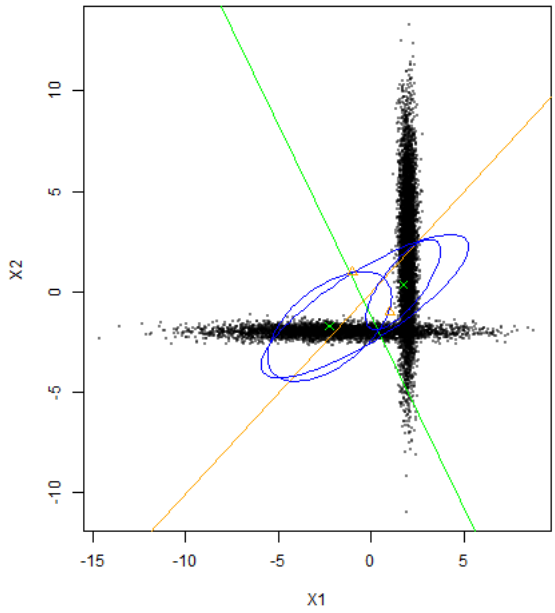

FIGURE 6.5. Illustration de l'adaptation, covariances étirées, $n = 200$. Paramètres de la cible : $\mu_1 = (-2, -2), \mu_2 = (2,2), \Sigma_1 = \text{diag}(10, 0.05), \Sigma_2 = \text{diag}(0.05,10), p = 0.4$. $t_0 = 100, T_c = 0.905, D_r = 0.0099$.

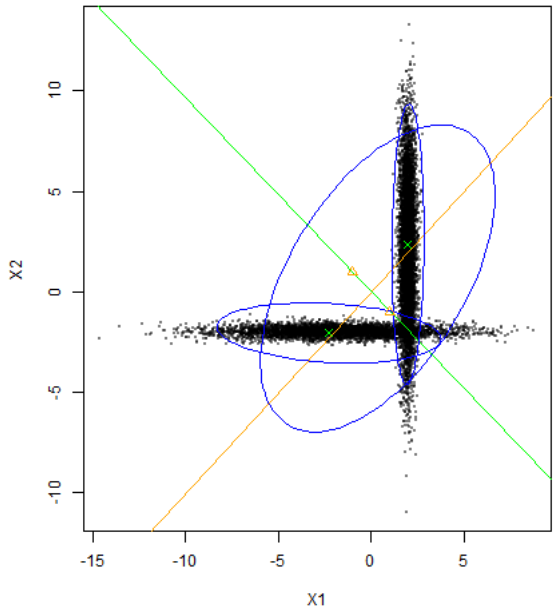

FIGURE 6.6. Illustration de l'adaptation, covariances étirées, $n = 2000$. Paramètres de la cible : $\mu_1 = (-2, -2), \mu_2 = (2,2), \Sigma_1 = \text{diag}(10, 0.05), \Sigma_2 = \text{diag}(0.05,10), p = 0.4$. $t_0 = 100, T_c = 0.9515, D_r = 0.24775$.

modes sont partiellement confondus, l'algorithme réussit à trouver une séparation linéaire satisfaisante pour la simulation.

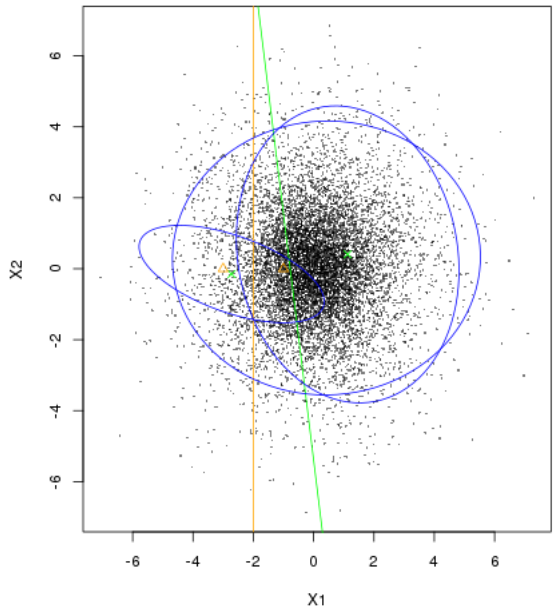

FIGURE 6.7. Illustration de l'adaptation, modes confondus, $n = 200$. Paramètres de la cible : $\mu_1 = (0,0), \mu_2 = (0,0) \Sigma_1 = I_2, \Sigma_2 = 4I_2, p = 0.5$. $t_0 = 100, T_c = 0.52, D_r = 0.633663$.

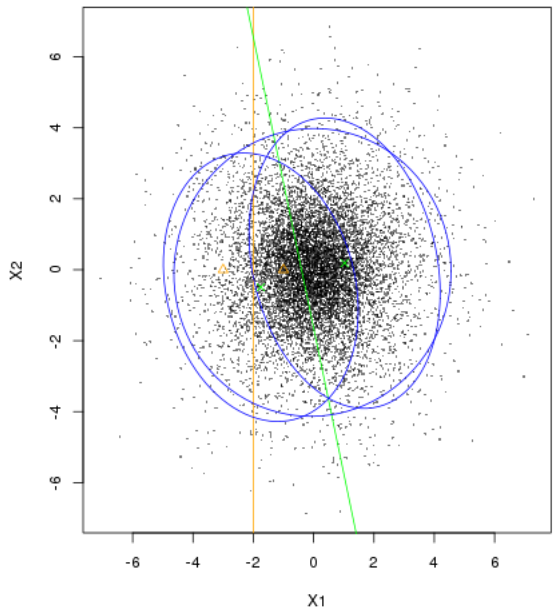

FIGURE 6.8. Illustration de l'adaptation, modes confondus, $n = 2000$. Paramètres de la cible : $\mu_1 = (0,0), \mu_2 = (0,0) \Sigma_1 = I_2, \Sigma_2 = 4I_2, p = 0.5$. $t_0 = 100, T_c = 0.54, D_r = 0.233766$.

Figures 6.5 et 6.6 : Voici un exemple d'une distribution relativement étirée et difficile, où l'algorithme parvient tout de même rapidement à déterminer la forme générale de la cible, ainsi qu'un plan intéressant.

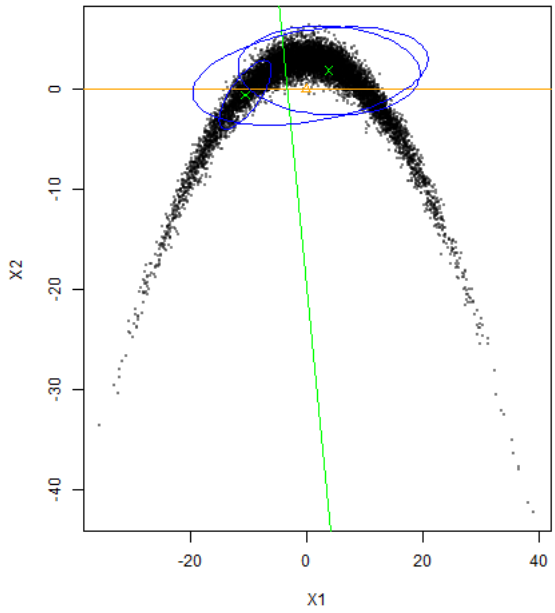

FIGURE 6.9. Illustration de l'adaptation, distribution irrégulière, $n = 200$. Paramètres de la cible : $\mu_1 = (0,0), \Sigma_1 = \text{diag}(100, 1), \psi = 0.03$. $t_0 = 100$.

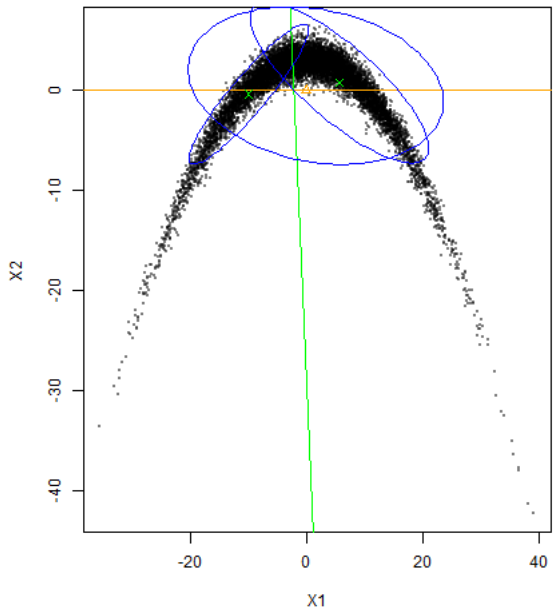

FIGURE 6.10. Illustration de l'adaptation, distribution irrégulière, $n = 2000$. Paramètres de la cible : $\mu_1 = (0,0), \Sigma_1 = \text{diag}(100, 1), \psi = 0.03$. $t_0 = 100$.

Figures 6.7 et 6.8 : Dans le cas où les modes sont confondus, ou pour des cibles unimodales avec symétrie sphérique, il n'y a pas vraiment de séparation optimale. Le plan finira tout de même par se stabiliser quelque part, mais il est clair que des algorithmes plus simples auront de meilleurs résultats pour de telles situations.

Figures 6.9 et 6.10 : Cette distribution irrégulière est suggérée par [15]. Nous voyons que même pour une distribution unimodale, un algorithme d'adaptation régionale peut engendrer des situations intéressantes, par exemple dans ce cas-ci une proposition adaptée à chaque branche de la distribution cible et une covariance globale permettant de passer de l'une à l'autre.

6.3. Tests comparatifs en faible dimension

Afin de comparer de façon aussi exhaustive que possible les capacités adaptatives à court terme des divers algorithmes à l'étude, nous rapportons les résultats d'un test de simulation très semblable à celui proposé dans [3]. Les 10 densités cibles choisies sont paramétrées de la façon suivante :

$$\mu_1 = -m(1,\ldots,1),\ \mu_2 = m(1,\ldots,1),$$
$$\Sigma_1 = I_d,\ \Sigma_2 = sI_d,\ p = 1/2,$$

où les paramètres d, m et s varient d'une cible à l'autre selon le tableau suivant :

	1	2	3	4	5	6	7	8	9	10
d	2	2	2	2	2	5	5	5	5	5
m	1	1	0	0	2	0.5	0.5	0	0	1
s	1	4	1	4	1	1	4	1	4	1

Dans tous les cas, les paramètres initiaux sont $n = 1000, t_0 = 100, k = 1$, $\mu_1^{(0)} = (-2,0,\ldots,0), \mu_2^{(0)} = (2,0,\ldots,0), C_1^{(0)} = I_d/10, C_2^{(0)} = sI_d/10$. De plus, $C_S^{(0)} = 50I_d$ si $d = 2$ et $C_S^{(0)} = 10I_d$ si $d = 5$. La partition initiale en deux régions est donnée par le plan $x_1 = 0$, ce qui s'avère une séparation assez bonne quand $d = 2$, mais beaucoup moins lorsque $d = 5$. Notons que ces choix favorisent légèrement l'algorithme RAPTOR, puisque l'initialisation inégale des covariances permet de définir pour cet algorithme une frontière initiale plus avantageuse que le plan $x_1 = 0$. La valeur initiale de la chaîne est l'origine $(0,\ldots,0)$. Chaque scénario de simulation a été répliqué 10^6 fois pour chacun de algorithmes.

Les tableaux 6. I et 6. II rapportent respectivement l'erreur quadratique moyenne empirique et la différence de couverture moyenne au niveau 95% multipliées par 1000, ainsi que les écarts-type estimés de ces estimateurs, pour chacun des cas de figure et chaque algorithme.

Globalement, on remarque tout d'abord que la qualité relative des algorithmes déterminée selon l'un ou l'autre des critères est assez différente. Alors que les mesures de taux de couvertures suggèrent que l'algorithme RAPTOR est préférable dans presque toutes les situations, les conclusions basées sur l'erreur quadratique par rapport à la vraie moyenne sont davantage mitigées. Tel que discuté auparavant, nous accorderons davantage d'importance au taux de couverture, mais nos conclusions seront d'autant plus fortes si elles sont

Algorithme	RAPT		Raptor		$OPRA_0$		$OPRA_1$		OPRA	
(d,m,s)	Moyenne	Écart-type	Moyenne	Écart-type	Moyenne	Écart-type	Moyenne	Écart-type	Moyenne	Écart-type
(2, 1, 1)	44.23	0.17	43.32	0.16	45.61	0.17	46.03	0.17	38.41	0.14
(2, 1, 4)	89.64	0.31	90.95	0.31	90.32	0.32	89.55	0.31	80.65	0.27
(2, 0, 1)	19.74	0.06	17.53	0.06	17.95	0.06	18.50	0.06	17.44	0.06
(2, 0, 4)	50.90	0.17	49.86	0.16	50.41	0.17	50.30	0.16	49.77	0.17
(2, 2, 1)	360.39	1.82	412.39	2.23	367.06	1.82	390.27	2.10	247.48	1.42
(5, 1/2, 1)	150.99	0.38	155.94	0.38	145.72	0.35	149.53	0.37	153.79	0.37
(5, 1/2, 4)	416.04	0.89	357.17	0.77	402.05	0.86	404.00	0.87	429.07	0.92
(5, 0, 1)	110.74	0.24	116.20	0.25	105.13	0.22	106.44	0.22	115.16	0.24
(5, 0, 4)	254.41	0.56	264.80	0.58	246.65	0.53	248.90	0.54	260.42	0.57
(5, 1, 1)	614.08	2.51	582.24	2.36	621.30	2.51	683.01	2.77	619.66	2.61

TABLEAU 6. I. Tests en basse dimension, erreur quadratique moyenne empirique fois 10^3.

Algorithme	RAPT		Raptor		$OPRA_0$		$OPRA_1$		OPRA	
(d,m,s)	Moyenne	Écart-type	Moyenne	Écart-type	Moyenne	Écart-type	Moyenne	Écart-type	Moyenne	Écart-type
(2, 1, 1)	3.17	0.05	1.95	0.05	2.88	0.05	2.83	0.06	2.95	0.06
(2, 1, 4)	3.84	0.06	3.02	0.06	3.59	0.06	3.43	0.06	3.61	0.06
(2, 0, 1)	3.68	0.06	1.08	0.06	3.34	0.05	3.72	0.06	2.39	0.06
(2, 0, 4)	5.93	0.06	1.73	0.06	5.06	0.06	5.12	0.06	3.19	0.06
(2, 2, 1)	-5.96	0.06	-4.42	0.06	-6.29	0.06	-6.02	0.06	-6.04	0.06
(5, 1/2, 1)	14.48	0.06	14.12	0.06	13.80	0.06	13.60	0.06	13.86	0.06
(5, 1/2, 4)	23.81	0.06	18.83	0.07	23.18	0.06	22.85	0.06	23.70	0.06
(5, 0, 1)	14.64	0.06	14.60	0.06	14.01	0.06	13.70	0.06	14.08	0.06
(5, 0, 4)	23.97	0.06	20.59	0.07	23.42	0.06	22.94	0.06	23.64	0.06
(5, 1, 1)	13.30	0.06	12.16	0.06	12.70	0.06	12.55	0.06	12.89	0.07

TABLEAU 6. II. Tests en basse dimension, différence de taux de couverture à 95%, fois 10^3.

corroborées par d'autres critères également. Par ailleurs, les cas où les deux critères sont en désaccord peuvent aussi apporter des informations intéressantes. Par exemple, dans le cas (5,0,4), l'algorithme RAPTOR semble explorer plus rapidement que les autres les queues de la distribution cible, mais au prix d'une convergence plus lente de l'estimation de la moyenne.

De façon générale, on observe que le RAPTOR semble faire mieux que les autres algorithmes pour un même nombre d'itérations, tel que prévu. Il a obtenu la différence de taux de couverture la plus faible dans 8 des 10 exemples, avec un écart très marqué dans 6 de ces derniers cas. À l'opposé, l'algorithme RAPT a offert les taux les moins satisfaisants à deux exceptions près. Finalement, les trois versions de notre algorithme ont offert généralement des performance intermédiaires entre ces deux extrêmes. Les valeurs obtenues suggèrent que leur efficacité est habituellement plus proche de celle du RAPT que de celle du RAPTOR.

Toutefois, notons que dans deux situations, nos algorithmes ont fait mieux que RAPTOR selon les deux critères. Dans le premier cas $(5,\frac{1}{2},1)$, nous avons une situation où les deux modes sont relativement bien séparés tout en n'étant pas trop distants, ce qui représente des circonstances assez idéales pour notre algorithme. En effet, une séparation linéaire entre les deux modes est alors amplement suffisante et risque de se stabiliser plus rapidement que la séparation du RAPTOR basée sur les densités empiriques. Dans le second cas (5, 0, 1), nous avons une distribution unimodale et il est difficile de déterminer ce qui a pu favoriser notre

algorithme. Il semble plus plausible que ce soit en fait une légère faiblesse systématique de RAPTOR en basse dimension, argumentée à la section 6.4.1, qui soit en cause ici.

À l'inverse, dans le cas (2,2,1), où les deux modes sont particulièrement éloignés l'un de l'autre, notre algorithme a fait pire que le RAPT. Cela est possiblement dû au fait qu'une première adaptation trop hâtive de l'hyperplan peut accentuer les problèmes de sous-représentation d'un mode. Ainsi, d'une part, le fait que le plan initial séparait déjà presque parfaitement les deux modes a donné un certain avantage au RAPT. D'autre part, cela suggère qu'il pourrait être utile d'intégrer à notre algorithme un critère de prolongement de la pré-adaptation lorsque l'une des deux régions estimées est trop peu explorée.

Les trois versions évaluées de notre algorithme ont présenté des performances comparables pour la plupart des cibles. La version pondérée s'est démarquée favorablement des deux autres dans deux cas unimodaux ((2,0,1) et (2,0,4)), une tendance que nous continuerons à explorer dans les simulations suivantes. Dans les autres cas, la version pondérée a fait légèrement moins bien que les autres, mais il faut tenir compte du fait que le temps de pré-adaptation très court de ces simulations est particulièrement nuisible à cette dernière. De même, bien que la version OPRA_1 avec re-calcul semble avoir donné des résultats un peu plus intéressants ici que la version de base, cette tendance ne s'est pas reproduite dans les tests en plus grande dimension.

Nous pouvons conclure de ces tests initiaux que notre algorithme réussit habituellement à s'adapter plus efficacement que l'algorithme RAPT à des cibles en basse dimension avec un temps de calcul marginalement plus élevé. Il ne fait en général pas aussi bien que le RAPTOR, mais peut surpasser ce dernier dans certaines situations, tout en étant nettement plus rapide pour un même nombre d'itérations, comme nous le verrons plus loin. Nous verrons aussi des tests ajustés en fonction du temps de calcul.

Rappelons que cette série de simulations avait pour but de vérifier la capacité d'adaptation à court terme des algorithmes, ce qui n'est qu'un aperçu partiel de leurs capacités. Nous nous tournons maintenant vers la partie centrale de notre investigation, soit le comportement à long terme de ceux-ci sur des cibles en grande dimension.

6.4. Tests comparatifs en grande dimension

Nous évaluons maintenant les capacités des algorithmes à reproduire fidèlement des densités cibles en grande dimension. Les tests sont assez similaires à ce que ces méthodes seront appelées à faire en pratique, avec l'avantage que nous connaissons la nature et les propriétés des distributions cibles, ce qui nous permet de mesurer précisément le niveau de convergence des processus. Nous présenterons en particulier neuf distributions cibles choisies de façon à illustrer des exemples typiques mais aussi un grand éventail de défis, incluant des cas problématiques pour un ou plusieurs des algorithmes comparés.

Toutes ces distributions ont été testées dans les quatre paramétrisations globales suivantes, choisies dans le but de pouvoir détecter des différences de comportement selon le nombre de dimensions ou le temps relatif de pré-adaptation.

d	n	t_0	N
5	10^4	10^3	10^4
20	10^5	10^3	10^4
20	10^5	10^4	10^4
50	$2 \cdot 10^5$	10^4	10^3

Dans bien des cas, les résultats sont assez analogues d'une paramétrisation à l'autre et nous avons alors choisi de présenter dans cette section uniquement les résultats en 50 dimensions, tout en conservant l'ensemble des résultats à l'annexe D.

Dans toutes les situations, nous avons eu recours à $k = 4$ chaînes en adaptation parallèle, tel que décrit à la section 2.4. Ceci a pour effet d'accroître la performance de tous les algorithmes par rapport à l'utilisation d'une seule chaîne et cet accroissement est relativement plus marqué pour le RAPT. Cet ajout ne présente aucun désavantage d'un point de vue numérique, bien au contraire, et il est donc tout indiqué de l'intégrer ici, de façon à refléter plus fidèlement ce qui peut se faire en pratique. Des tests préliminaires semblent indiquer que d'augmenter la valeur de k au-delà de 4 ne modifie pas les performances relatives des algorithmes l'un par rapport à l'autre, nous nous en sommes donc tenus à cette seule valeur.

De façon générale, nous avons utilisé des conditions initiales en accord avec ce qui est fait dans la littérature, soit des matrices de covariance $C_1^{(0)}$ et $C_2^{(0)}$ de taille réduite (au sens de la norme) qui favorisent un taux d'acceptation élevé au sein de chaque mode durant la pré-adaptation, et une matrice de covariance globale $C_S^{(0)}$ de taille relativement grande, permettant de couvrir convenablement l'espace d'intérêt.

Nous présentons maintenant les résultats pour chacune des cibles d'intérêt, en débutant par les plus simples.

6.4.1. Cibles unimodales simples

Le premier exemple est une simple normale unimodale sphérique, avec une délimitation initiale appropriée (voir tableau 6. III). Sans surprise, bien que tous les algorithmes présentent des résultats intéressants, RAPTOR, étant particulièrement apte à s'adapter à des distribution unimodales, obtient les meilleurs taux de couvertures. On remarque qu'il présente les valeurs les moins intéressantes pour les deux critères de performance secondaires, un phénomène que nous reverrons à plusieurs reprises. De même, les taux d'acceptation indiquent que le comportement de RAPTOR est fondamentalement différent de celui des autres algorithmes. Par contre, un coup d'oeil au tableau complet en annexe indique aussi qu'il est

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
$d = 50 \quad n = 2 \cdot 10^5 \quad t_0 = 10^4 \quad N = 10^3$								
Temps (s)	32.9151	2.7978e-02	44.7172	6.0209e-02	33.8018	1.3089e-01	33.4221	1.2294e-02
EQM_e	0.0423	2.7973e-04	0.0476	4.8417e-04	0.0424	2.7086e-04	0.0443	2.9995e-04
AQV	1.2795	1.9896e-04	1.2405	3.7615e-03	1.2797	1.8900e-04	1.2769	1.9712e-04
Couv. 50% (%)	6.5054	3.7036e-02	5.5101	6.6740e-02	6.4996	3.7972e-02	5.5313	3.8994e-02
Couv. 90% (%)	2.4718	1.7961e-02	2.0276	2.4707e-02	2.4738	1.7698e-02	2.1804	1.8600e-02
Couv. 95% (%)	1.3921	1.1705e-02	1.1447	1.5350e-02	1.4025	1.1504e-02	1.2397	1.2011e-02
Couv. 99% (%)	0.3328	4.0638e-03	0.2756	5.0512e-03	0.3445	4.1654e-03	0.3054	4.3408e-03
T_a	0.2533	5.6501e-05	0.2782	7.7208e-04	0.2535	5.7450e-05	0.2528	9.6509e-05

TABLEAU 6. III. Résultats partiels, cible normale unimodale sphérique : $\mu_1 = (0, \ldots, 0), \Sigma_1 = I_d$. Paramètres : $k = 4, \mu_1^{(0)} = X_0^{(1)} = X_0^{(2)} = (-0.1{,}0, \ldots, 0), \mu_2^{(0)} = X_0^{(3)} = X_0^{(4)} = (0.1{,}0, \ldots, 0), C_1^{(0)} = C_2^{(0)} = 0.1 I_d, C_S^{(0)} = 2 I_d$, Plan initial : $x_1 = 0$.

en fait le plus faible des algorithmes dans le scénario à cinq dimensions, un phénomène qui se reproduit dans de nombreux autres exemples.

D'autre part, RAPT et OPRA_0 présentent des performances assez semblables, avec un léger avantage pour RAPT. Ceci est certainement dû au fait que l'adaptation de l'hyperplan n'apporte ici aucun avantage et qu'elle provoque plutôt une volatilité supplémentaire dans l'estimation des covariances, ce qui ralentit l'adaptation des autres paramètres. Malgré cela, il n'y a pas de désavantage marqué si on utilise OPRA_0 plutôt que RAPT dans cette situation.

De façon étonnante, la version pondérée d'OPRA présente des taux de couverture moyens très intéressants qui le rapprochent davantage du RAPTOR que des autres algorithmes. Un examen plus détaillé démontre qu'en fait, la flexibilité supplémentaire dans l'adaptation de l'hyperplan a souvent pour effet d'éliminer essentiellement l'une des régions artificielles. Autrement dit, toute la région d'intérêt par rapport à la cible finit par se retrouver d'un même côté de l'hyperplan, et le processus reste presque toujours pris de ce bon côté de l'hyperplan. Cela rend la distribution instrumentale pratiquement unimodale, donc beaucoup mieux adaptée à la distribution cible.

On observe un phénomène semblable pour le RAPTOR, c'est-à-dire que dans la plupart des simulations, l'un des deux modes de la densité instrumentale finit par se voir attribuer un poids très proche de 1, alors que l'autre mode n'est plus vraiment utilisé. Dans les deux algorithmes, il semble que ce soit un déséquilibre dans l'exploration des deux régions empiriques initiales, amplifié par la flexibilité d'adaptation de la séparation, qui mène à l'élimination progressive d'un des modes. Un tel déséquilibre devient bien sûr de plus en plus probable à mesure que la dimension augmente. Ceci pourrait aussi expliquer pourquoi les performances de RAPTOR sont moins intéressantes dans le cas en cinq dimensions. En effet, nous observons que l'élimination d'un mode y est beaucoup plus rare, probablement parce que l'exploration initiale est en général assez équilibrée. La plus grande capacité de

RAPTOR introduirait alors une volatilité inutile dans l'adaptation, sans apporter d'avantage particulier en retour.

Avant de poursuivre, nous émettrons quelques commentaires sur les temps de calcul, puisque nous pouvons d'ores et déjà observer certains phénomènes qui se reproduisent dans toutes les simulations à venir. Le tableau 6. IV résume les temps moyens observés pour tous les scénarios.

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Scénario	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
$d = 5$	0.0258	6.6393e-06	0.0362	4.7901e-06	0.0264	9.5570e-06	0.0276	5.3412e-06
$d = 20$ $t_0 = 10^3$	2.3071	1.4761e-03	2.9216	8.3671e-04	2.3061	1.1711e-03	2.3712	1.4231e-03
$d = 20$ $t_0 = 10^4$	2.1226	1.9644e-03	2.9814	5.4297e-03	2.1092	3.9597e-04	2.4452	5.7275e-03
$d = 50$	32.9151	2.7978e-02	44.7172	6.0209e-02	33.8018	1.3089e-01	33.4221	1.2294e-02

TABLEAU 6. IV. Temps de calcul, cible normale unimodale sphérique : $\mu_1 = (0, \ldots, 0), \Sigma_1 = I_d$. Paramètres : $k = 4, \mu_1^{(0)} = X_0^{(1)} = X_0^{(2)} = (-5/d, 0, \ldots, 0), \mu_2^{(0)} = X_0^{(3)} = X_0^{(4)} = (5/d, 0, \ldots, 0), C_1^{(0)} = C_2^{(0)} = 0.1 I_d, C_S^{(0)} = 100 I_d/d$, Plan initial : $x_1 = 0$.

On remarque d'emblée que l'algorithme RAPTOR prend de loin le plus de temps pour compléter ses itérations, dans un rapport d'environ 4/3 par rapport à RAPT ou OPRA_0, et ce, indépendamment de la cible ou de la dimension. Les autres algorithmes ont des temps de calcul habituellement plus comparables entre eux, avec très rarement des temps anormalement plus longs pour OPRA par rapport aux deux autres.

On observe aussi que l'ordre naturel attendu, soit RAPT $<$ OPRA_0 $<$ OPRA, est perturbé de temps à autre. Ceci peut s'expliquer par le fait que le temps de calcul est partiellement influencé par la réalisation particulière du processus simulé. En effet, lorsqu'à une étape donnée, le candidat proposé ne se trouve pas dans la même région que le point actuel du processus, le calcul du seuil d'acceptation nécessite l'évaluation de plusieurs densités normales supplémentaires (voir (3.2.6)).

Il est tout à fait envisageable que certains algorithmes aient tendance à favoriser plus que d'autres la proposition d'un candidat dans une même région. Par exemple, comme nous l'avons mentionné plus tôt, en 50 dimensions, l'algorithme OPRA a tendance à écarter l'une des deux régions, faisant en sorte que la plupart des candidats proposés se trouvent dans la même région que la valeur actuelle. Ceci réduit significativement le nombre de calculs à effectuer, au point où cet algorithme devient alors en moyenne plus rapide qu'OPRA_0, malgré une adaptation plus coûteuse.

Finalement, notons que les simulations pour un scénario donné ont été réalisées au même moment sur quatre processeurs parallèles d'une même machine, un par algorithme. Les temps d'une même ligne peuvent donc être comparés de façon assez fiable. Toutefois, il pourrait s'avérer trompeur de comparer les temps mesurés sur différentes densités cibles. Également,

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
			$d = 5 \quad n = 10^4 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	0.0337	1.6032e-05	0.0486	1.7488e-05	0.0345	1.3860e-05	0.0364	1.4967e-05
EQM_e	0.7198	8.4064e-03	0.6702	7.5415e-03	0.7299	8.4658e-03	0.6346	7.2945e-03
AQV	12.9881	1.1079e-02	16.0990	1.4929e-02	12.8885	1.0929e-02	15.6640	1.2811e-02
Couv. 50% (%)	1.9723	1.9489e-02	2.2938	1.9824e-02	1.9610	1.9439e-02	1.9385	1.9651e-02
Couv. 90% (%)	0.7723	1.0859e-02	0.9309	1.0858e-02	0.7566	1.0922e-02	0.7463	1.0730e-02
Couv. 95% (%)	0.4463	7.4455e-03	0.5356	7.4378e-03	0.4325	7.5203e-03	0.4283	7.3877e-03
Couv. 99% (%)	0.1133	2.9809e-03	0.1347	2.9913e-03	0.1099	3.0243e-03	0.1077	3.0126e-03
T_a	0.3470	9.1819e-05	0.3549	1.1216e-04	0.3471	9.1531e-05	0.3441	1.0925e-04
			$d = 50 \quad n = 2 \cdot 10^5 \quad t_0 = 10^4 \quad N = 10^3$					
Temps (s)	42.8289	5.6423e-02	56.7632	9.9536e-02	42.8397	6.5971e-02	43.4046	5.2014e-02
EQM_e	0.5137	1.8581e-02	0.6840	2.4418e-02	0.5752	1.8383e-02	0.3473	1.2701e-02
AQV	2.3931	3.0999e-03	2.6304	1.5650e-02	2.3871	2.9990e-03	2.8513	4.5295e-03
Couv. 50% (%)	6.8135	3.9836e-02	6.1825	1.0193e-01	6.8060	3.9577e-02	6.0684	3.8504e-02
Couv. 90% (%)	2.6015	1.7822e-02	2.2466	3.4089e-02	2.5821	1.8229e-02	2.3439	1.8784e-02
Couv. 95% (%)	1.4746	1.1516e-02	1.2693	1.9562e-02	1.4678	1.1838e-02	1.3414	1.1992e-02
Couv. 99% (%)	0.3594	4.3492e-03	0.3046	5.5356e-03	0.3554	4.2448e-03	0.3274	4.3504e-03
T_a	0.2546	6.1770e-05	0.2716	1.3502e-03	0.2545	5.9445e-05	0.2549	1.0906e-04

TABLEAU 6. V. Résultats partiels, cible normale unimodale étirée : $\mu_1 = (0,\dots,0), \Sigma_1 = \text{diag}(100,1,\dots,1)$. Paramètres : $k = 4, \mu_1^{(0)} = X_0^{(1)} = X_0^{(2)} = (5 - 5/d,0,\dots,0), \mu_2^{(0)} = X_0^{(3)} = X_0^{(4)} = (5 + 5/d,0,\dots,0), C_1^{(0)} = C_2^{(0)} = 0.1I_d, C_S^{(0)} = 100I_d/d$, Plan initial : $x_1 = 5$.

l'implémentation que nous avons choisie effectue systématiquement des calculs qui sont parfois seulement utiles pour OPRA et RAPTOR (voir annexe C). Ceci augmente donc artificiellement le temps de calcul de RAPT et OPRA_0, mais les ordres de grandeur observés demeurent tout de même utiles et le principal élément à retenir est que RAPTOR est significativement plus lent que ses compétiteurs.

La cible suivante (voir tableau 6. V) est une autre normale univariée, mais cette fois-ci avec une forme étirée selon la première composante, et une délimitation initiale sous-optimale. La plupart des observations faites lors de l'exemple précédent sont encore valables. Remarquons de plus que cette situation est volontairement plus ardue pour l'algorithme RAPT. Nous remarquons en conséquence des taux de couverture légèrement meilleurs pour OPRA_0. Par ailleurs, les estimations de la moyenne et l'AQV de l'algorithme OPRA sont particulièrement intéressantes ici.

6.4.2. Cibles unimodales complexes

Dans cette section, nous étudions les résultats des algorithmes sur deux versions de la distribution dite *banane*, provenant de [15], et représentée sous sa forme bidimensionnelle dans les figures 6.9 et 6.10. Le premier cas est celui où la torsion est particulièrement prononcée, mais où la partition initiale est aussi bonne que possible. Les résultats du tableau 6. VI indiquent que cette cible est effectivement plus difficile que les précédentes, et qu'il est

très difficile en grandes dimensions d'équilibrer les deux branches pour obtenir une bonne estimation de la moyenne, ainsi que d'explorer adéquatement les queues de la distribution. Nous constatons sans surprise que l'algorithme OPRA_0 est assez semblable à l'algorithme RAPT ici selon tous les critères mesurés. L'algorithme RAPTOR demeure de loin préférable et l'algorithme OPRA offre encore une fois le meilleur second choix, quoique son estimation de la moyenne soit relativement moins intéressante ici.

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
	$d = 50$	$n = 2 \cdot 10^5$	$t_0 = 10^4$	$N = 10^3$				
Temps (s)	32.8377	2.8269e-02	43.6053	3.4642e-02	32.8733	2.1112e-02	33.2980	7.0866e-03
EQM_e	33.3458	2.2474e-01	27.2430	2.4339e-01	33.3240	2.2415e-01	36.1961	2.3272e-01
AQV	1.3758	1.0702e-03	1.2198	4.7294e-03	1.3674	8.8923e-04	1.3427	1.1385e-03
Couv. 50% (%)	10.2879	4.4969e-02	8.7374	8.2639e-02	10.1773	4.5731e-02	9.4461	4.9672e-02
Couv. 90% (%)	3.6125	1.8525e-02	3.0517	3.0105e-02	3.5817	1.8687e-02	3.3109	2.0693e-02
Couv. 95% (%)	1.9492	1.2037e-02	1.6191	1.8120e-02	1.9251	1.2174e-02	1.7722	1.3585e-02
Couv. 99% (%)	0.3585	4.4101e-03	0.2677	5.8793e-03	0.3493	4.4195e-03	0.3109	5.0854e-03
T_a	0.1935	2.7006e-04	0.1741	5.9588e-04	0.1925	2.6917e-04	0.1788	3.4516e-04

TABLEAU 6. VI. Résultats comparatifs partiels, cible normale unimodale étirée avec torsion : $\mu_1 = (0,\ldots,0), \Sigma_1 = \text{diag}(100,1,\ldots,1), \psi = 0.1$. Paramètres : $k = 4, X_0^{(1)} = \cdots = X_0^{(4)} = (0,\ldots,0), \mu_1^{(0)} = (-0.1,0,\ldots,0), \mu_2^{(0)} = (0.1,0,\ldots,0), C_1^{(0)} = C_2^{(0)} = 0.1 I_d, C_S^{(0)} = I_d$, Plan initial : $x_1 = 0$.

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
	$d = 50$	$n = 2 \cdot 10^5$	$t_0 = 10^4$	$N = 10^3$				
Temps (s)	42.7070	2.8671e-02	56.8099	6.4008e-02	42.8216	2.7467e-02	43.5063	2.4836e-02
EQM_e	39.5713	2.4068e-01	27.2430	2.4339e-01	33.8121	2.2890e-01	36.2137	2.2480e-01
AQV	1.3316	1.2390e-03	1.2198	4.7294e-03	1.3512	7.9258e-04	1.3448	1.0822e-03
Couv. 50% (%)	11.5616	5.3307e-02	8.7374	8.2639e-02	11.2031	4.7012e-02	10.6810	4.5114e-02
Couv. 90% (%)	3.9098	2.0397e-02	3.0517	3.0105e-02	3.7471	1.9164e-02	3.5902	1.9755e-02
Couv. 95% (%)	2.0874	1.2820e-02	1.6191	1.8120e-02	2.0066	1.2164e-02	1.9170	1.2739e-02
Couv. 99% (%)	0.3796	4.6791e-03	0.2677	5.8793e-03	0.3693	4.3782e-03	0.3476	4.8216e-03
T_a	0.1986	6.0065e-04	0.1741	5.9588e-04	0.1997	2.9903e-04	0.1824	3.4859e-04

TABLEAU 6. VII. Résultats comparatifs partiels, cible normale unimodale étirée avec torsion et mauvaise partition initiale : $\mu_1 = (0,\ldots,0), \Sigma_1 = \text{diag}(100,1,\ldots,1), \psi = 0.1$. Paramètres : $k = 4, X_0^{(1)} = \cdots = X_0^{(4)} = (0,\ldots,0), \mu_1^{(0)} = (-0.1,0,\ldots,0), \mu_2^{(0)} = (0.1,0,\ldots,0), C_1^{(0)} = C_2^{(0)} = 0.1 I_d, C_S^{(0)} = I_d$, Plan initial : $x_2 = 0$.

Dans cette première situation, la délimitation initiale $x_1 = 0$ séparait exactement les deux branches de la distribution cible, un hyperplan qui se voulait optimal. Le tableau 6. VII résume le cas où la séparation initiale $x_2 = 0$ occasionne en principe des distributions instrumentales moins appropriées dans chaque régions. Alors que l'algorithme RAPTOR n'est pas du tout affecté par cette modification, les trois autres algorithmes sont ralentis par

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
$d = 20 \quad n = 10^5 \quad t_0 = 10^4 \quad N = 10^4$								
Temps (s)	3.0404	1.9644e-03	4.1112	2.3451e-03	3.0537	1.9707e-03	3.1497	2.1761e-03
EQM_e	1.3454	9.3948e-03	1.2225	1.0183e-02	1.1482	9.4891e-03	1.2554	9.9476e-03
AQV	3.0526	1.3472e-03	2.5037	1.2190e-03	2.3393	1.2094e-03	2.5585	1.1256e-03
Couv. 50% (%)	2.9929	1.5111e-02	1.8152	1.5628e-02	2.8886	1.4020e-02	2.8977	1.4305e-02
Couv. 90% (%)	1.4157	7.5302e-03	0.8730	8.0200e-03	1.3573	6.9461e-03	1.3610	7.1302e-03
Couv. 95% (%)	0.8471	4.9529e-03	0.5245	5.3087e-03	0.8115	4.5666e-03	0.8115	4.7530e-03
Couv. 99% (%)	0.2252	1.8161e-03	0.1393	1.9669e-03	0.2144	1.7112e-03	0.2147	1.7808e-03
T_a	0.1921	1.0275e-04	0.2232	1.1184e-04	0.2073	1.0007e-04	0.1965	9.9635e-05
$d = 50 \quad n = 2 \cdot 10^5 \quad t_0 = 10^4 \quad N = 10^3$								
Temps (s)	44.2777	2.7295e-02	56.8833	1.0425e-01	44.4561	3.1596e-02	45.0652	2.7542e-02
EQM_e	2.1909	2.6101e-02	2.1137	4.1433e-02	1.8272	3.1483e-02	2.1315	3.6926e-02
AQV	2.2010	2.7381e-03	1.5037	1.2587e-02	1.7234	1.4714e-03	1.8650	1.8174e-03
Couv. 50% (%)	9.7818	4.4858e-02	8.8190	1.0926e-01	9.0352	4.1220e-02	8.5845	4.2168e-02
Couv. 90% (%)	3.5478	1.8397e-02	3.2784	3.8886e-02	3.3034	1.8247e-02	3.1433	1.8652e-02
Couv. 95% (%)	1.9820	1.1724e-02	1.8567	2.2348e-02	1.8642	1.1469e-02	1.7732	1.1897e-02
Couv. 99% (%)	0.4712	4.1492e-03	0.4514	6.1624e-03	0.4487	4.0829e-03	0.4221	4.2818e-03
T_a	0.2266	2.8329e-04	0.2063	1.4120e-03	0.2311	1.8157e-04	0.2233	1.9109e-04

TABLEAU 6. VIII. Résultats comparatifs partiels, cible normale unimodale étirée avec torsion légère et mauvaise partition initiale : $\mu_1 = (0,\ldots,0), \Sigma_1 = \text{diag}(100,1,\ldots,1), \psi = 0.03$. Paramètres : $k = 4, X_0^{(1)} = \cdots = X_0^{(4)} = (0,\ldots,0), \mu_1^{(0)} = (-0.1,0,\ldots,0), \mu_2^{(0)} = (0.1,0,\ldots,0), C_1^{(0)} = C_2^{(0)} = 0.1 I_d, C_S^{(0)} = I_d$, Plan initial : $x_2 = 0$.

cette nouvelle condition. Ce ralentissement est particulièrement marqué pour RAPT, qui n'a aucune façon d'établir une meilleure partition de l'espace. Ces résultats illustrent donc bien dans quelle mesure l'algorithme OPRA_0 peut se distinguer avantageusement du RAPT dans des cas particulièrement défavorables à ce dernier. On peut s'attendre à des résultats intermédiaires entre ces extrêmes pour des partitions initiales qui seraient mieux choisies mais sans être optimales.

Finalement, le tableau 6. VIII résume un cas où le degré de torsion appliqué sur la distribution cible est plus faible, un cas donc plus facile en principe. De façon étonnante, alors qu'en dimension modérée les choses semblent se passer comme dans les cas précédents, en 50 dimensions, l'algorithme RAPTOR n'est plus le choix optimal. Il est plutôt comparable à l'algorithme OPRA_0 qui offre de son côté une meilleure estimation de la moyenne, alors que c'est l'algorithme OPRA qui offre les meilleurs taux de couverture. En fait, les taux de couverture de RAPTOR ne sont pas plus intéressants ici que dans les cas précédents avec torsion plus prononcée.

De façon globale, ces trois cas apparentés représentent des situations où les différentes versions de notre algorithme font mieux que RAPT, tout en nécessitant des temps de calcul supplémentaires inférieurs à 2% sur les exemples en 50 dimensions. Pour un nombre

d'itérations égal, l'algorithme RAPTOR demeure généralement supérieur, quoiqu'il puisse démontrer de la faiblesse sur certains exemples.

6.4.3. Cibles bimodales simples

Nous passons maintenant à des cibles bimodales, l'application visée des méthodes que nous étudions. Dans plusieurs des cas étudiés, la partition initiale est de moins en moins optimale à mesure que la dimension augmente, ce qui est une bonne chose puisqu'en pratique, il sera plus ardu de déterminer *a priori* une bonne partition en grande dimension.

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
			$d = 20 \quad n = 10^5 \quad t_0 = 10^4 \quad N = 10^4$					
Temps (s)	2.6773	2.3826e-03	3.4153	2.8216e-03	2.6489	1.9922e-03	2.7280	2.4052e-03
EQM_e	0.0143	4.5711e-05	0.0180	5.8945e-05	0.0141	4.5348e-05	0.0142	4.5349e-05
AQV	1.2280	8.5275e-05	1.2094	1.0404e-04	1.2276	8.5732e-05	1.2276	8.5768e-05
Couv. 50% (%)	1.0284	1.1414e-02	0.4425	1.2436e-02	1.0278	1.1457e-02	0.8834	1.1607e-02
Couv. 90% (%)	0.4639	6.1957e-03	0.1250	6.7600e-03	0.4654	6.2191e-03	0.3984	6.2686e-03
Couv. 95% (%)	0.3134	4.1895e-03	0.1033	4.5395e-03	0.3151	4.2078e-03	0.2748	4.2271e-03
Couv. 99% (%)	0.0818	1.6270e-03	0.0185	1.7491e-03	0.0810	1.6317e-03	0.0697	1.6483e-03
T_a	0.2660	2.4858e-05	0.2901	4.6193e-05	0.2661	2.4865e-05	0.2642	3.3089e-05
T_c	0.5785	8.6396e-05	0.5189	9.3477e-04	0.6250	4.5049e-04	0.6603	1.2882e-04
D_r	0.3256	2.1704e-04	0.3245	2.4707e-04	0.3252	2.2016e-04	0.3257	2.1533e-04
TC_e	0.0308	8.6697e-06	0.0349	1.3609e-04	0.0320	9.3115e-06	0.0081	8.0497e-06

TABLEAU 6. IX. Résultats complets, cible normale bimodale : $\mu_1 = -(2.5,\ldots,2.5)/d, \mu_2 = (2.5,\ldots,2.5)/d, \Sigma_1 = \Sigma_2 = I_d, p = 0.6$. Paramètres : $k = 4, \mu_1^{(0)} = X_0^{(1)} = X_0^{(2)} = (-2,0,\ldots,0), \mu_2^{(0)} = X_0^{(3)} = X_0^{(4)} = (2,0,\ldots,0), C_1^{(0)} = C_2^{(0)} = 0.1I_d, C_S^{(0)} = 100I_d/d$, Plan initial : $x_1 = 0$.

Dans la plupart des cas simples, où les modes sont relativement bien séparés sans être complètement distincts, et où les paramètre initiaux sont raisonnables sans être idéaux, on observe un comportement similaire à ce qui est présenté dans le tableau 6. IX. On remarque d'entrée de jeu que tous les algorithmes mènent à de bons estimés de la moyenne et qu'ils sont très stables selon toutes les mesures employées, comme en témoignent les écarts-types minimes. L'algorithme RAPTOR est encore une fois de loin préférable, bien que ses estimations de la moyenne convergent souvent plus lentement. Les performances relatives des autres algorithmes sont également conformes à ce que nous avons observé en général jusqu'à maintenant.

Tel que prévu, les deux versions d'OPRA réussissent à améliorer le taux de concordance entre les régions estimées et les véritables régions basé sur la densité maximale des modes. Malgré cela, l'algorithme RAPT fait ici aussi bien que OPRA_0, en raison de la flexibilité apportée par les autres paramètres adaptatifs, mais aussi à cause de la forme similaire des deux modes. Dans ce cas facile donc, l'utilisation d'OPRA_0 n'apporte pas d'avantage marqué,

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
			$d = 20 \quad n = 10^5 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	3.5274	8.4242e-04	4.6488	1.2748e-03	3.5551	9.2663e-04	3.6259	7.1877e-04
EQM_e	0.0476	1.7047e-04	0.0457	1.5714e-04	0.0468	1.6869e-04	0.0466	1.6738e-04
AQV	2.4235	1.7088e-03	2.3560	1.4991e-03	2.4187	1.6735e-03	2.4438	1.6934e-03
Couv. 50% (%)	8.4275	4.3952e-02	6.9969	4.1908e-02	8.1722	4.3395e-02	7.6128	4.4279e-02
Couv. 90% (%)	2.6274	1.0584e-02	2.1086	1.0595e-02	2.5264	1.0615e-02	2.3948	1.0937e-02
Couv. 95% (%)	1.4387	5.8126e-03	1.1532	5.9495e-03	1.3829	5.8590e-03	1.3156	6.0309e-03
Couv. 99% (%)	0.3340	1.6314e-03	0.2664	1.7597e-03	0.3189	1.6818e-03	0.3068	1.7083e-03
T_a	0.2582	1.0063e-04	0.3032	1.1536e-04	0.2656	1.0404e-04	0.2616	1.1015e-04
T_c	0.6460	2.5920e-04	0.5197	7.7894e-04	0.5682	5.2386e-04	0.5826	4.3911e-04
D_r	0.1772	8.4553e-04	0.1500	7.8216e-04	0.1723	8.2953e-04	0.1622	8.3121e-04
TC_e	0.0209	1.1917e-05	0.0003	2.0756e-06	0.0294	1.6614e-05	0.0038	8.9876e-06
			$d = 20 \quad n = 10^5 \quad t_0 = 10^4 \quad N = 10^4$					
Temps (s)	3.2816	8.7595e-04	4.4251	1.1006e-03	3.3036	6.4243e-04	3.3830	9.6434e-04
EQM_e	0.0469	1.6011e-04	0.0468	1.5853e-04	0.0467	1.5782e-04	0.0462	1.5608e-04
AQV	2.4664	1.8771e-03	2.4503	1.8790e-03	2.4615	1.8928e-03	2.4648	1.8545e-03
Couv. 50% (%)	4.7755	4.8483e-02	3.9636	4.7799e-02	4.8115	4.8908e-02	4.6081	4.8211e-02
Couv. 90% (%)	1.6873	1.2174e-02	1.3545	1.2295e-02	1.6893	1.2334e-02	1.5971	1.2239e-02
Couv. 95% (%)	0.9519	6.7313e-03	0.7600	6.8407e-03	0.9529	6.7765e-03	0.9007	6.7472e-03
Couv. 99% (%)	0.2311	1.9206e-03	0.1856	1.9778e-03	0.2318	1.9158e-03	0.2198	1.9127e-03
T_a	0.2443	1.1417e-04	0.2903	1.5668e-04	0.2467	1.1976e-04	0.2521	1.4303e-04
T_c	0.6269	2.8400e-04	0.5085	4.3139e-04	0.6331	3.2161e-04	0.5491	4.1159e-04
D_r	0.1177	7.7934e-04	0.1067	7.2161e-04	0.1187	7.8493e-04	0.1151	7.6899e-04
TC_e	0.0203	1.2419e-05	0.0101	8.5761e-05	0.0249	1.6789e-05	0.0127	1.6094e-05
			$d = 50 \quad n = 2 \cdot 10^5 \quad t_0 = 10^4 \quad N = 10^3$					
Temps (s)	52.4107	3.8650e-01	62.5131	4.4145e-01	53.9307	4.1217e-01	53.7983	3.9853e-01
EQM_e	0.1476	8.8943e-04	0.2263	1.5408e-03	0.1437	8.7343e-04	0.1457	9.1248e-04
AQV	1.4753	5.8478e-03	0.7249	7.0816e-03	1.5071	6.7895e-03	1.5152	7.0460e-03
Couv. 50% (%)	40.8048	2.3946e-01	46.6756	1.2665e-01	39.4484	2.7311e-01	39.1409	2.7492e-01
Couv. 90% (%)	8.9612	3.1543e-02	9.8757	7.0676e-03	8.7431	3.7089e-02	8.6908	3.8618e-02
Couv. 95% (%)	4.5493	1.4396e-02	4.9574	2.8174e-03	4.4487	1.6978e-02	4.4229	1.7929e-02
Couv. 99% (%)	0.9296	2.6513e-03	0.9954	5.3062e-04	0.9133	3.1448e-03	0.9067	3.4489e-03
T_a	0.2623	4.3011e-04	0.1668	1.4387e-03	0.2660	4.8399e-04	0.2621	5.2826e-04
T_c	0.8054	1.3656e-03	0.6044	7.9911e-03	0.6167	1.2799e-03	0.8440	2.4368e-03
D_r	0.8134	4.7761e-03	0.9322	2.5418e-03	0.7857	5.4451e-03	0.7820	5.4104e-03
TC_e	0.0117	2.7427e-05	0.0010	5.1141e-06	0.0200	3.1313e-05	0.0046	3.2504e-05

TABLEAU 6. X. Résultats partiels, cible normale bimodale avec variances inégales : $\mu_1 = -(2.5, \ldots, 2.5)/d, \mu_2 = (2.5, \ldots, 2.5)/d, \Sigma_1 = 4I_d, \Sigma_2 = I_d, p = 0.6)$. Paramètres : $k = 4, \mu_1^{(0)} = X_0^{(1)} = X_0^{(2)} = (-2,0,\ldots,0), \mu_2^{(0)} = X_0^{(3)} = X_0^{(4)} = (2,0,\ldots,0), C_1^{(0)} = C_2^{(0)} = 0.1I_d, C_S^{(0)} = 100I_d/d$, Plan initial : $x_1 = -1$.

mais ne requiert pas beaucoup plus de calcul non plus. Finalement, RAPTOR présente un taux de concordance plutôt bas, et en général celui-ci ne dépasse guère 60 %.

Le tableau 6. X résume un cas où les covariances de chaque mode sont inégales, au point où les deux modes sont confondus en une seule masse de densité non-symétrique. En 20 dimensions, les choses se passent de façon habituelle, quoiqu'il soit intéressant de remarquer qu'une plus longue période de pré-adaptation favorise ici davantage l'algorithme RAPT que

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
$d = 50 \quad n = 2 \cdot 10^5 \quad t_0 = 5 \cdot 10^4 \quad N = 10^3$								
Temps (s)	48.6759	1.6913e-01	56.4119	2.7428e-01	48.6783	1.5268e-01	49.1709	1.6715e-01
EQM_e	0.1642	1.1463e-03	0.2902	1.7465e-03	0.1627	1.1973e-03	0.1637	1.1690e-03
AQV	1.7986	1.3812e-02	0.5985	4.3974e-03	1.8295	1.4834e-02	1.8452	1.5560e-02
Couv. 50% (%)	31.3102	3.6205e-01	43.9261	1.9831e-01	30.6486	3.8556e-01	30.0736	4.0245e-01
Couv. 90% (%)	7.2071	6.1492e-02	9.3660	2.4145e-02	7.1011	6.4265e-02	6.9404	6.9628e-02
Couv. 95% (%)	3.7104	2.9844e-02	4.7148	1.1618e-02	3.6664	3.0877e-02	3.5759	3.3728e-02
Couv. 99% (%)	0.7794	5.8388e-03	0.9505	2.6903e-03	0.7722	6.1109e-03	0.7535	6.6124e-03
T_a	0.2427	6.7317e-04	0.2055	9.2315e-04	0.2446	6.4573e-04	0.2453	7.5195e-04
T_c	0.5171	4.8501e-04	0.7114	5.0607e-03	0.5143	6.2390e-04	0.6572	3.8885e-03
D_r	0.5339	7.8595e-03	0.8065	5.8245e-03	0.5204	8.3386e-03	0.5182	8.2657e-03
TC_e	0.0182	3.2599e-05	0.0053	1.2637e-05	0.0192	3.0238e-05	0.0091	3.3609e-05

TABLEAU 6. XI. Résultats supplémentaires, cible normale bimodale avec variances inégales : $\mu_1 = -(2.5,\ldots,2.5)/d, \mu_2 = (2.5,\ldots,2.5)/d, \Sigma_1 = 4I_d, \Sigma_2 = I_d, p = 0.6$). Paramètres : $k = 4, \mu_1^{(0)} = X_0^{(1)} = X_0^{(2)} = (-2,0,\ldots,0), \mu_2^{(0)} = X_0^{(3)} = X_0^{(4)} = (2,0,\ldots,0), C_1^{(0)} = C_2^{(0)} = 0.1I_d, C_S^{(0)} = 100I_d/d$, Plan initial : $x_1 = -1$.

l'algorithme OPRA_0. Ceci peut s'expliquer entre autres par l'impossibilité de trouver un hyperplan adaptatif basé uniquement sur les moyennes empiriques qui soit substantiellement meilleur que la délimitation initiale.

On remarque également que le cas en 50 dimensions est particulièrement difficile, et que c'est une autre situation où RAPTOR perd sa supériorité en faveur de l'algorithme OPRA. À première vue, la meilleure explication serait que le temps de pré-adaptation est insuffisant pour que celui-ci réussisse à détecter régulièrement le second mode, faisant en sorte que l'une des composantes est écartée. Or, en faisant des simulations avec un plus long temps de pré-adaptation, on observe que les performances de RAPTOR ne sont pas vraiment différentes, alors que celles des autres algorithmes sont nettement améliorées. Le tableau 6. XI montre que même avec $t_0 = 50000$, RAPTOR ne peut combler son écart par rapport aux autres choix.

Bien sûr, ce cas demeure difficile pour toutes les méthodes et il serait plus juste de dire que les autres font moins pire. La pré-adaptation requise par une telle cible paraît donc particulièrement longue. Or, les résultats obtenus suggèrent que les trois autres algorithmes sont plus tolérants à une trop courte pré-adaptation combinée à des conditions initiales potentiellement de faible qualité.

Il semblerait donc que nous ayons affaire à une forme de surajustement à la pré-adaptation affectant seulement l'algorithme RAPTOR. Ceci peut s'avérer problématique en pratique. En effet, dans bien des situations, il est impossible de déterminer quelle serait une valeur suffisante pour t_0. Un algorithme étant sensible à ce paramètre est donc difficile à gérer, puisqu'il n'y pas de façon infaillible en pratique de vérifier la qualité de la pré-adaptation.

6.4.4. Cibles bimodales complexes

Nous présentons enfin brièvement deux cibles ardues pour la plupart des algorithmes. La première est une normale bimodale où les moyennes des deux modes coïncident, similaire à l'exemple bidimensionnel représenté aux figures 6.7 et 6.8. Alors que cette situation devrait être invariablement à l'avantage de RAPTOR, ce qui est effectivement le cas en 20 dimensions, le tableau 6. XII indique que ce dernier éprouve une fois encore des difficultés en 50 dimensions, du moins relativement aux trois autres algorithmes qui offrent tous des performances à peu près identiques. Il semble qu'ici encore, l'algorithme RAPTOR ait tendance à mettre de côté l'un des deux modes, mais à tort cette fois-ci.

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
$d = 50 \quad n = 2 \cdot 10^5 \quad t_0 = 10^4 \quad N = 10^3$								
Temps (s)	34.3879	3.5111e-02	42.7624	5.4941e-02	34.4780	4.4372e-02	34.9097	3.6428e-02
EQM_e	0.0553	5.1254e-04	0.1081	1.2292e-03	0.0565	5.4570e-04	0.0576	5.4985e-04
AQV	1.4404	5.2328e-03	0.7839	7.6903e-03	1.4476	5.4267e-03	1.4376	5.6263e-03
Couv. 50% (%)	30.4250	1.9586e-01	35.9864	1.3119e-01	30.1101	2.0204e-01	30.0034	2.0183e-01
Couv. 90% (%)	8.7921	3.8640e-02	9.8361	8.8781e-03	8.7585	3.9779e-02	8.8580	4.0274e-02
Couv. 95% (%)	4.4742	1.7541e-02	4.9454	3.4774e-03	4.4632	1.7875e-02	4.5063	1.8202e-02
Couv. 99% (%)	0.9165	3.2711e-03	0.9938	7.0621e-04	0.9174	3.2254e-03	0.9234	3.2454e-03
T_a	0.2615	3.8586e-04	0.1792	1.5699e-03	0.2616	4.0890e-04	0.2562	4.4286e-04
T_c	0.5002	5.0639e-04	0.5434	8.6171e-03	0.5006	9.9433e-04	0.9002	2.1555e-03
D_r	0.8404	4.4720e-03	0.9359	2.4240e-03	0.8363	4.6109e-03	0.8449	4.6196e-03
TC_e	0.0192	2.1400e-05	0.0010	5.0895e-06	0.0210	2.4149e-05	0.0025	1.9568e-05

TABLEAU 6. XII. Résultats partiels, cible normale bimodale avec $\mu_1 = \mu_2 = (0, \ldots, 0)$;$\Sigma_1 = I_d, \Sigma_2 = 4I_d, p = 0.6$. Paramètres : $k = 4, \mu_1^{(0)} = X_0^{(1)} = X_0^{(2)} = (-2{,}0, \ldots, 0), \mu_2^{(0)} = X_0^{(3)} = X_0^{(4)} = (2{,}0, \ldots, 0), C_1^{(0)} = C_2^{(0)} = 0.1I_d, C_S^{(0)} = 100I_d/d$, Plan initial : $x_1 = 0$.

Finalement, le tableau 6. XIII résume le cas où la densité de l'un des modes est concentrée dans une région relativement petite. Il s'agit d'un des rares cas où une modification du temps de pré-adaptation change de façon marquée les performances relatives des algorithmes. Comme tous sont initialisés avec les mêmes conditions de générateurs de nombres pseudo-aléatoires, et comme les paramètres initiaux de RAPTOR induisent la même séparation que celle utilisée par les autres algorithmes, les comportements en pré-adaptation sont identiques. Ces exemples, surtout lorsque $t_0 = 10^3$, mesurent donc assez directement à quel point chaque algorithme parvient à se remettre d'une pré-adaptation trop courte n'ayant pas situé l'un des modes.

À ce chapitre, l'algorithme OPRA semble le mieux s'en sortir. Lorsque $t_0 = 10^4$, un temps plus raisonnable pour une cible de cette difficulté, RAPT prend l'avantage, en vertu de la bonne séparation initiale utilisée, alors que l'algorithme RAPTOR présente le même genre de retard que ceux vus précédemment. Notons qu'il est possible que les conditions initiales utilisées ici, par exemple le choix des valeurs de départ $\mu_1^{(0)}$ et $\mu_2^{(0)}$, désavantagent

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
			$d = 20 \quad n = 10^5 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	4.1971	9.3595e-03	5.4945	1.1813e-02	4.3097	9.9728e-03	4.4335	1.0314e-02
EQM_e	0.0976	6.5276e-04	0.0889	6.1257e-04	0.0991	6.6136e-04	0.0945	6.6780e-04
AQV	0.2491	1.5184e-03	0.2316	1.1243e-03	0.2527	1.5812e-03	0.2670	1.6896e-03
Couv. 50% (%)	18.0435	1.4054e-01	17.6445	1.2031e-01	18.3338	1.4180e-01	16.7202	1.4983e-01
Couv. 90% (%)	6.1841	3.7034e-02	6.0183	3.2444e-02	6.2623	3.7566e-02	5.7592	4.0134e-02
Couv. 95% (%)	3.2045	1.7991e-02	3.1084	1.5860e-02	3.2445	1.8257e-02	2.9913	1.9571e-02
Couv. 99% (%)	0.6691	3.5976e-03	0.6465	3.2065e-03	0.6768	3.6425e-03	0.6262	3.9157e-03
T_a	0.2246	3.8068e-04	0.2331	4.1139e-04	0.2104	4.0079e-04	0.1969	3.9693e-04
T_c	0.6346	2.3040e-04	0.5074	2.8952e-03	0.7492	2.0536e-03	0.7972	1.7366e-03
D_r	0.6434	2.7847e-03	0.6278	2.5956e-03	0.6489	2.8063e-03	0.6192	2.9210e-03
TC_e	0.0258	4.4824e-05	0.0094	1.1466e-04	0.0185	9.8921e-05	0.0008	6.3768e-06
			$d = 20 \quad n = 10^5 \quad t_0 = 10^4 \quad N = 10^4$					
Temps (s)	3.3849	2.8740e-03	4.5431	3.4228e-03	3.3959	2.8051e-03	3.4669	2.9281e-03
EQM_e	0.0604	5.9282e-04	0.0649	5.7734e-04	0.0611	5.9369e-04	0.0616	5.8570e-04
AQV	0.3786	2.0276e-03	0.3266	1.7702e-03	0.3761	2.0220e-03	0.3837	2.0673e-03
Couv. 50% (%)	5.3423	1.6149e-01	8.2444	1.5547e-01	5.8016	1.6122e-01	5.6138	1.6228e-01
Couv. 90% (%)	2.4845	4.6374e-02	3.3460	4.4243e-02	2.5736	4.6517e-02	2.4824	4.6914e-02
Couv. 95% (%)	1.3761	2.2993e-02	1.7796	2.1964e-02	1.4216	2.3068e-02	1.3672	2.3265e-02
Couv. 99% (%)	0.3069	4.7495e-03	0.3787	4.5743e-03	0.3142	4.7667e-03	0.3011	4.8160e-03
T_a	0.1885	3.9549e-04	0.2190	4.5458e-04	0.1856	3.9498e-04	0.1752	4.0655e-04
T_c	0.6196	2.8266e-04	0.5318	1.3115e-03	0.7450	9.9762e-04	0.6952	1.7175e-03
D_r	0.4160	2.7965e-03	0.4702	2.7867e-03	0.4249	2.7859e-03	0.4210	2.8202e-03
TC_e	0.0203	3.9415e-05	0.0226	1.1879e-04	0.0197	5.7978e-05	0.0079	2.7194e-05

TABLEAU 6. XIII. Résultats complets, cible normale bimodale avec singularité : $\mu_1 = -(2.5,\ldots,2.5)/d, \mu_2 = (2.5,\ldots,2.5)/d, \Sigma_1 = I_d/10, \Sigma_2 = I_d, p = 0.6$. Paramètres : $k = 4, \mu_1^{(0)} = X_0^{(1)} = X_0^{(2)} = (-2,0,\ldots,0), \mu_2^{(0)} = X_0^{(3)} = X_0^{(4)} = (2,0,\ldots,0), C_1^{(0)} = C_2^{(0)} = 0.1I_d, C_S^{(0)} = 100I_d/d$, Plan initial : $x_1 = 0$.

ce dernier. Ces résultats n'en demeurent pas moins importants, puisqu'en pratique il ne sera pas toujours évident de déterminer des valeurs initiales appropriées.

6.5. Résultats ajustés pour le temps

Afin de tenir compte du temps réel d'exécution des algorithmes, certains tests ont été effectués avec un nombre inégal d'itérations pour chaque algorithme, en proportion inverse de leur vitesse anticipée de calcul. Le tableau 6. XIV résume certains de ces tests.

Bien que les temps ne soient pas exactement égaux, l'exercice est suffisant pour montrer que l'efficacité théorique supérieure de RAPTOR ne se traduit pas nécessairement en une vitesse de convergence accrue en temps réel. Dans ce cas précis, c'est OPRA qui semble préférable par une petite marge, mais la généralisation de tels résultats est difficile étant donné que les temps d'exécution relatifs dépendent de l'implémentation particulière utilisée et surtout de la nature de la densité cible. Par exemple, certaines cibles sont beaucoup plus complexes à calculer qu'une densité normale bimodale, ce qui rend le temps dédié à l'adaptation assez négligeable par rapport à celui dédié au calcul du rapport de densités.

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
n	100000		80000		99000		97000	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
			$d = 20 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	3.5271	8.2836e-04	3.7412	9.2687e-04	3.5383	8.4477e-04	3.5091	1.0230e-03
EQM_e	0.0519	1.8743e-04	0.0647	2.2500e-04	0.0516	1.8518e-04	0.0526	1.8695e-04
AQV	2.8086	1.8817e-03	2.6191	1.7487e-03	2.8019	1.8290e-03	2.8090	1.8002e-03
Couv. 50% (%)	8.1694	3.9052e-02	8.3758	4.0249e-02	7.9263	3.7967e-02	7.5908	3.8666e-02
Couv. 90% (%)	2.4042	9.5828e-03	2.3832	1.0185e-02	2.3344	9.3398e-03	2.2516	9.5919e-03
Couv. 95% (%)	1.2849	5.4215e-03	1.2674	5.8797e-03	1.2500	5.3420e-03	1.2077	5.4599e-03
Couv. 99% (%)	0.2934	1.6233e-03	0.2877	1.8362e-03	0.2876	1.6275e-03	0.2779	1.6611e-03
T_a	0.2656	9.6584e-05	0.3210	1.2024e-04	0.2720	1.0377e-04	0.2732	1.0729e-04
			$d = 20 \quad t_0 = 10^4 \quad N = 10^4$					
Temps (s)	3.2735	7.4111e-04	3.4890	8.8166e-04	3.2762	7.2445e-04	3.2829	8.0101e-04
EQM_e	0.0509	1.6986e-04	0.0649	2.1529e-04	0.0518	1.7417e-04	0.0524	1.7481e-04
AQV	2.8732	1.9997e-03	2.7532	2.1937e-03	2.8648	2.0193e-03	2.8516	2.0079e-03
Couv. 50% (%)	4.3640	4.2312e-02	4.7586	4.7017e-02	4.4089	4.2857e-02	4.3919	4.2830e-02
Couv. 90% (%)	1.4751	1.0572e-02	1.5065	1.1978e-02	1.4760	1.0802e-02	1.4441	1.0960e-02
Couv. 95% (%)	0.8050	6.0257e-03	0.8133	6.8629e-03	0.8044	6.1241e-03	0.7836	6.2863e-03
Couv. 99% (%)	0.1940	1.8210e-03	0.1921	2.0882e-03	0.1953	1.8588e-03	0.1878	1.8949e-03
T_a	0.2525	1.0353e-04	0.3093	1.5279e-04	0.2545	1.0685e-04	0.2608	1.3453e-04

TABLEAU 6. XIV. Résultats avec itérations ajustées pour le temps, cible normale bimodale avec variances inégales : $\mu_1 = -(2.5,\ldots,2.5)/d, \mu_2 = (2.5,\ldots,2.5)/d, \Sigma_1 = I_d/10, \Sigma_2 = I_d, p = 0.6$. Paramètres : $k = 4, \mu_1^{(0)} = X_0^{(1)} = X_0^{(2)} = (-2,0,\ldots,0), \mu_2^{(0)} = X_0^{(3)} = X_0^{(4)} = (2,0,\ldots,0), C_1^{(0)} = C_2^{(0)} = 0.1I_d, C_S^{(0)} = 100I_d/d$, Plan initial : $x_1 = 0$.

De tels cas seraient alors sans doute à l'avantage de RAPTOR, à tout le moins en faisant abstraction des difficultés occasionnelles déjà mentionnées pour cet algorithme.

6.6. RÉSUMÉ

Ce chapitre a exploré, sous de nombreux aspects, les forces et faiblesses des algorithmes avec adaptation régionale mis à l'étude. Nous prenons ici quelques lignes pour faire le point sur les résultats présentés et insister sur les principaux éléments relevés dans cette analyse.

– L'algorithme RAPTOR est incontestablement supérieur en dimension modérée. Il explore plus rapidement que les autres algorithmes les queues des distributions cibles, au prix d'une convergence un peu plus lente d'estimateurs à tendance centrale tels que la moyenne. En petite dimension, sa complexité supplémentaire est souvent inutile, et parfois même contre-productive. En grande dimension, il est habituellement le meilleur choix, mais peut éprouver des difficultés avec des distributions cibles particulièrement complexes. En général, ceci semble causé par un phénomène de sur-ajustement à l'échantillon prélevé lors de la pré-adaptation, menant à une plus grande incapacité à récupérer de la non-détection d'un mode lors de la dite pré-adaptation.

– L'algorithme OPRA fait significativement mieux que l'algorithme RAPT dans presque tous les exemples considérés, incluant de nombreuses cibles unimodales, et n'a pas été affecté par les problèmes rapportés pour RAPTOR. Son temps de calcul est relativement plus élevé que celui d'OPRA$_0$, mais bien inférieur à celui de RAPTOR.
– Nous avons vérifié graphiquement que l'algorithme OPRA$_0$ se comporte tel que prévu au point de vue de l'adaptation. Il a des performances similaire à l'algorithme RAPT sur les cas où la partition initiale de l'espace est assez bonne. Dans les situations moins idéales, il converge légèrement plus rapidement. Dans tous les cas, son temps de calcul est à peine plus long que celui du RAPT.
– Basé seulement sur le nombre d'itérations, l'algorithme RAPTOR présente habituellement les meilleurs résultats. Par contre, en nous basant sur les temps réels d'exécution, pour certaines densités cibles constituées d'un mélange de deux normales, nous avons vu que cet avantage peut s'estomper complètement, au point d'en faire le choix le moins efficace parmi ceux considérés.
– Il est à noter que les temps de calculs sont bien sûr influencés par la complexité de la densité cible, puisqu'un rapport de densité doit être calculé à chaque itération. Dans la plupart des applications, c'est en fait ce calcul du rapport de densité qui consomme la grande majorité du temps d'exécution. Les écarts de temps relatifs entre les algorithmes que nous avons observés seraient donc moins prononcés dans la plupart des utilisations réelles. Ces résultats demeurent néanmoins utiles dans une perspective d'optimiser le compromis entre l'efficacité en temps réel et l'efficacité en nombre d'itérations des processus adaptatifs.

Chapitre 7

CONCLUSION

Ce mémoire avait pour but d'étudier les propriétés théoriques et le comportement empirique de certaines méthodes MCMC avec adaptation régionale, incluant un algorithme que nous avons développé. Nous avons d'abord présenté un survol général de ces méthodes basées sur l'algorithme original de Metropolis-Hastings. Nous avons ensuite détaillé le fonctionnement de deux algorithmes avec adaptation régionale bien connus, soient le RAPT et le RAPTOR, pour ensuite expliciter notre algorithme OPRA et ses variantes, qui sont essentiellement des extensions de l'algorithme RAPT où nous avons ajouté une composante permettant l'adaptation de la frontière linéaire délimitant les deux régions échantillonnales.

Par la suite, nous avons présenté et justifié de façon détaillée des conditions suffisantes de convergence pour les algorithmes adaptatifs. Nous avons utilisé celles-ci afin de démonter l'ergodicité de notre algorithme, en nous basant sur des arguments géométriques et sur une preuve similaire d'ergodicité pour l'algorithme RAPT, preuve que nous avons également incluse, détaillée et généralisée aux cas à plus de deux régions. Nous avons également fourni une démonstration de l'ergodicité des processus à chaînes parallèles qui sont souvent utilisés dans des contextes adaptatifs.

Finalement, nous avons présenté des mesures de performance pour les MCMC à cible bien déterminée, que nous avons utilisées afin de comparer les différents algorithmes, avec des résultats parfois surprenants. Entre autres, nous avons découvert que l'algorithme RAPTOR n'est pas systématiquement supérieur à ses compétiteurs moins flexibles, sans même tenir compte de sa plus grande complexité de calcul, et que des mesures doivent parfois être prises pour minimiser les conséquences d'une mauvaise pré-adaptation. Nous avons aussi remarqué que la version la plus simple de notre algorithme, OPRA_0, est somme toute assez similaire à l'algorithme RAPT en terme de performance et de temps de calcul. Au contraire, OPRA, la version utilisant l'information des covariances empiriques dans l'adaptation de son hyperplan, a montré des résultats très intéressants, considérant que celle-ci est limitée à des séparations linéaires.

De ces observations, nous pouvons dégager certaines recommandations pratiques. Premièrement, il n'y a aucun désavantage substantiel à ajouter de façon permanente la composante de base OPRA_0 à l'algorithme RAPT, et cet ajout peut mener à des gains d'efficacité intéressants lorsque la partition empirique originale de l'espace est très sous-optimale. En basse dimension, disons $d < 5$, l'algorithme RAPTOR est généralement à éviter, à moins d'avoir de bonnes raisons de croire que la forme de la densité cible est particulièrement complexe. Dans les cas plus fréquents où le nombre de dimensions est élevé, l'algorithme RAPTOR est en général le meilleur choix, quoique par une moins grande marge qu'on pourrait le penser une fois considérée la vitesse de convergence en temps réel. De plus, en présence d'une cible à première vue assez complexe, il faudra songer à des méthodes pour éviter un surajustement à des conditions initiales ou une période de pré-adaptation sous-optimales. Alternativement, on pourrait considérer l'algorithme OPRA, qui démontre une bonne efficacité et paraît robuste aux problèmes mentionnés pour RAPTOR.

Il serait donc tout naturel d'envisager des procédures hybrides combinant d'une façon ou d'une autre ces deux algorithmes. Par exemple, les méthodes à chaînes parallèles que nous avons présentées et utilisées peuvent tout aussi bien fonctionner avec différents algorithmes dans un même processus. Ceci pourrait permettre d'absorber la faiblesse relative de l'un ou l'autre des schémas d'adaptation sur un cas particulier et de réduire la probabilité de ne pas détecter un mode de la distribution cible. On pourrait d'autre part envisager une adaptation en deux temps, où l'algorithme OPRA serait utilisé en début de processus pour déterminer une paramétrisation approximative intéressante, pour ensuite céder la place à l'algorithme RAPTOR qui peaufinerait l'adaptation, tout en évitant certains problèmes liés à la pré-adaptation parfois observés pour cet algorithme. Finalement, il serait intéressant d'étudier comment ces méthodes peuvent se combiner à d'autres stratégies cherchant à résoudre les problèmes de multimodalité, notamment les MCMC à propositions multiples, tout en gardant à l'esprit le souci de simplicité qui nous a incité a développer la méthode alternative proposée dans ce mémoire.

BIBLIOGRAPHIE

[1] Andrieu, C. et J. Thoms. 2008, «A tutorial on adaptive MCMC», *Stat. Comput.*, vol. 18, n° 4, doi :10.1007/s11222-008-9110-y, p. 343–373, ISSN 0960-3174. URL http://dx.doi.org/10.1007/s11222-008-9110-y.

[2] Atchadé, Y. F. et J. S. Rosenthal. 2005, «On adaptive Markov chain Monte Carlo algorithms», *Bernoulli*, vol. 11, n° 5, doi :10.3150/bj/1130077595, p. 815–828, ISSN 1350-7265. URL http://dx.doi.org/10.3150/bj/1130077595.

[3] Bai, Y., R. V. Craiu et A. F. Di Narzo. 2011, «Divide and conquer : a mixture-based approach to regional adaptation for MCMC», *J. Comput. Graph. Statist.*, vol. 20, n° 1, doi :10.1198/jcgs.2010.09035, p. 63–79, ISSN 1061-8600. URL http://dx.doi.org/10.1198/jcgs.2010.09035, supplementary material available online.

[4] Bédard, M., R. Douc et E. Moulines. 2012, «Scaling analysis of multiple-try MCMC methods», *Stochastic Process. Appl.*, vol. 122, n° 3, doi :10.1016/j.spa.2011.11.004, p. 758–786, ISSN 0304-4149. URL http://dx.doi.org/10.1016/j.spa.2011.11.004.

[5] Brooks, S. P. et A. Gelman. 1998, «General methods for monitoring convergence of iterative simulations», *J. Comput. Graph. Statist.*, vol. 7, n° 4, doi :10.2307/1390675, p. 434–455, ISSN 1061-8600. URL http://dx.doi.org/10.2307/1390675.

[6] Craiu, R. V., J. Rosenthal et C. Yang. 2009, «Learn from thy neighbor : parallel-chain and regional adaptive MCMC», *J. Amer. Statist. Assoc.*, vol. 104, n° 488, doi :10.1198/jasa.2009.tm08393, p. 1454–1466, ISSN 0162-1459. URL http://dx.doi.org/10.1198/jasa.2009.tm08393.

[7] Dempster, A. P., N. M. Laird et D. B. Rubin. 1977, «Maximum likelihood from incomplete data via the EM algorithm», *J. Roy. Statist. Soc. Ser. B*, vol. 39, n° 1, p. 1–38, ISSN 0035-9246. With discussion.

[8] Drummond, A. J. et A. Rambaut. 2007, «"beast : Bayesian evolutionary analysis by sampling trees», *BMC Evol Biol*, vol. 7, n° 3, p. 214. URL http://doi.org/10.1186/1471-2148-7-214.

[9] Flegal, J. M. et G. L. Jones. 2011, «Implementing MCMC : estimating with confidence», dans *Handbook of Markov chain Monte Carlo*, Chapman & Hall/CRC Handb. Mod. Stat. Methods, CRC Press, Boca Raton, FL, p. 175–197.

[10] Gelman, A., G. O. Roberts et W. R. Gilks. 1996, «Efficient Metropolis jumping rules», dans *Bayesian statistics, 5 (Alicante, 1994)*, Oxford Sci. Publ., Oxford Univ. Press, New York, p. 599–607.

[11] Geweke, J. 1992, «Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments», dans *IN BAYESIAN STATISTICS*, University Press, p. 169–193.

[12] Gilks, W. R., S. Richardson et D. J. Spiegelhalter, éd.. 1996, *Markov chain Monte Carlo in practice*, Interdisciplinary Statistics, Chapman & Hall, London, ISBN 0-412-05551-1, xviii+486 p.. URL https://doi.org/10.1007/978-1-4899-4485-6.

[13] Guan, Y. et S. M. Krone. 2007, «Small world MCMC and convergence to multi-modal distributions : from slow mixing to fast mixing», *Ann. Appl. Probab.*, vol. 17, n° 1, doi : 10.1214/105051606000000772, p. 284–304, ISSN 1050-5164. URL http://dx.doi.org/10.1214/105051606000000772.

[14] Haario, H., M. Laine, A. Mira et E. Saksman. 2006, «DRAM : efficient adaptive MCMC», *Stat. Comput.*, vol. 16, n° 4, doi :10.1007/s11222-006-9438-0, p. 339–354, ISSN 0960-3174. URL http://dx.doi.org/10.1007/s11222-006-9438-0.

[15] Haario, H., E. Saksman et J. Tamminen. 1999, «Adaptive proposal distribution for random walk metropolis algorithm», *Computational Statistics*, vol. 14, n° 3, doi :10.1007/s001800050022, p. 375–395, ISSN 0943-4062. URL https://doi.org/10.1007/s001800050022.

[16] Haario, H., E. Saksman et J. Tamminen. 2001, «An adaptive metropolis algorithm», *Bernoulli*, vol. 7, n° 2, doi :10.2307/3318737, p. 223–242, ISSN 1350-7265. URL http://dx.doi.org/10.2307/3318737.

[17] Haario, H., E. Saksman et J. Tamminen. 2005, «Componentwise adaptation for high dimensional MCMC», *Comput. Statist.*, vol. 20, n° 2, doi :10.1007/BF02789703, p. 265–273, ISSN 0943-4062. URL http://dx.doi.org/10.1007/BF02789703.

[18] Hastie, T., R. Tibshirani et J. Friedman. 2009, *The elements of statistical learning*, 2e éd., Springer Series in Statistics, Springer, New York, ISBN 978-0-387-84857-0, xxii+745 p.. URL https://doi.org/10.1007/978-0-387-84858-7, data mining, inference, and prediction.

[19] Hastings, W. K. 1970, «Monte carlo sampling methods using markov chains and their applications», *Biometrika*, vol. 57, n° 1, p. pp. 97–109, ISSN 00063444. URL http://www.jstor.org/stable/2334940.

[20] Heidelberger, P. et P. D. Welch. 1983, «Simulation run length control in the presence of an initial transient», *Operations Research*, vol. 31, n° 6, doi :10.1287/opre.31.6.1109, p. 1109–1144. URL https://doi.org/10.1287/opre.31.6.1109.

[21] Horn, R. A. et C. R. Johnson. 2013, *Matrix analysis*, 2e éd., Cambridge University Press, Cambridge, ISBN 978-0-521-54823-6, xviii+643 p..

[22] Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller et E. Teller. 1953, «Equation of state calculations by fast computing machines», *J. Chem. Phys.*, vol. 21, p. 1087.

[23] Meyn, S. P. et R. L. Tweedie. 1993, *Markov chains and stochastic stability*, Communications and Control Engineering Series, Springer-Verlag London, Ltd., London, ISBN 3-540-19832-6, xvi+548 p., doi :10.1007/978-1-4471-3267-7. URL http://dx.doi.org/10.1007/978-1-4471-3267-7.

[24] Mira, A. 2001, «On Metropolis-Hastings algorithms with delayed rejection», *Metron*, vol. 59, n° 3-4, p. 231–241 (2002), ISSN 0026-1424.

[25] Raftery, A. E. et S. Lewis. 1992, «How many iterations in the gibbs sampler ?», dans *In Bayesian Statistics 4*, Oxford University Press, p. 763–773.

[26] Roberts, G. O. et J. S. Rosenthal. 2001, «Optimal scaling for various Metropolis-Hastings algorithms», *Statist. Sci.*, vol. 16, n° 4, p. 351–367, ISSN 0883-4237. URL https://doi.org/10.1214/ss/1015346320.

[27] Roberts, G. O. et J. S. Rosenthal. 2004, «General state space Markov chains and MCMC algorithms», *Probab. Surv.*, vol. 1, doi :10.1214/154957804100000024, p. 20–71, ISSN 1549-5787. URL http://dx.doi.org/10.1214/154957804100000024.

[28] Roberts, G. O. et J. S. Rosenthal. 2007, «Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms», *J. Appl. Probab.*, vol. 44, n° 2, doi :10.1239/jap/1183667414, p. 458–475, ISSN 0021-9002. URL http://dx.doi.org/10.1239/jap/1183667414.

[29] Roberts, G. O. et J. S. Rosenthal. 2009, «Examples of adaptive MCMC», *J. Comput. Graph. Statist.*, vol. 18, n° 2, doi :10.1198/jcgs.2009.06134, p. 349–367, ISSN 1061-8600. URL http://dx.doi.org/10.1198/jcgs.2009.06134.

[30] Sahu, S. K. et A. A. Zhigljavsky. 2003, «Self-regenerative Markov chain Monte Carlo with adaptation», *Bernoulli*, vol. 9, n° 3, doi :10.3150/bj/1065444811, p. 395–422, ISSN 1350-7265. URL http://dx.doi.org/10.3150/bj/1065444811.

[31] Solonen, A., P. Ollinaho, M. Laine, H. Haario, J. Tamminen et H. Järvinen. 2012, «Efficient MCMC for climate model parameter estimation : parallel adaptive chains and early rejection», *Bayesian Anal.*, vol. 7, n° 3, doi :10.1214/12-BA724, p. 715–736, ISSN 1936-0975. URL http://dx.doi.org/10.1214/12-BA724.

Annexe A

RÉSULTATS THÉORIQUES INTERMÉDIAIRES

Nous démontrons ici certains résultats intermédiaires utilisés dans les démonstrations d'ergodicité. Ceux-ci concernent essentiellement les propriétés des matrices de covariance adaptatives C_t et leur lien avec la densité normale. Les résultats sont divisés par sujet et présentés autant que possible dans l'ordre de leur utilisation dans le chapitre 4.

A.1. Matrices : bornes et convergence

A.1.1. Théorèmes préliminaires

Commençons par citer quelques théorèmes d'algèbre linéaire concernant les matrices réelles symétriques. Ceux-ci s'étendent facilement aux matrices hermitiennes en général, mais cette généralisation ne nous sera pas utile ici. Pour la démonstration de ces énoncés et pour plus de résultats sur les sujets élaborés dans cette annexe, on pourra se référer aux chapitres 4 et 7 de [21].

Théorème A.1.1 (Diagonalisation de matrices symétriques)**.** *Soit A une matrice réelle symétrique. Alors, toutes ses valeurs propres sont réelles et il existe une matrice orthogonale Q ($QQ^T = Q^TQ = Q$) telle que $A = Q\Lambda Q^T$, où Λ est une matrice diagonale contenant chacune des valeurs propres de A.*

Théorème A.1.2 (Weil (fragment))**.** *Soient A et B, des matrices symétriques de taille d et notons par $\lambda_i(A), \lambda_i(B), \lambda_i(A+B),\ i \in \{1, \ldots, d\}$ les valeurs propres respectives de A, B et $A+B$ ordonnées respectivement en ordre croissant. Alors, pour $1 \leq j \leq i \leq d$:*

$$\lambda_{i-j+1}(A) + \lambda_j(B) \leq \lambda_i(A+B).$$

En particulier, pour tout i,

$$\lambda_i(A) + \lambda_i(B) \leq \lambda_i(A+B).$$

Théorème A.1.3 (Rayleigh (fragment))**.** *Soit A une matrice symétrique et λ_1 et λ_d, ses valeurs propre minimale et maximale respectivement. Alors,*

$$\lambda_1 = \min_{x \neq 0} \frac{x^T A x}{x^T x} \text{ et } \lambda_d = \max_{x \neq 0} \frac{x^T A x}{x^T x}.$$

A.1.2. Matrices définies positives

Plusieurs des propriétés intéressantes des matrices C_t découlent du fait qu'elles sont définies positives. Nous rappelons la définition de cette propriété et des résultats qui en découlent.

Définition A.1.1 (Matrice définie positive)**.** *Une matrice à coefficient réels symétrique A est dite définie positive si pour tout vecteur $x \neq 0$ de dimension appropriée,*

$$x^T A x > 0.$$

Il existe plusieurs caractérisations équivalentes à cette définition, nous en utiliserons seulement une autre ici.

Proposition A.1.1 (Caractérisation par les valeurs propres)**.** *Une matrice symétrique A est définie positive si et seulement si toutes ses valeurs propres sont strictement positives.*

Démonstration. Vérifions chacune des implications.

1. Soit λ, une valeur propre quelconque de A définie positive, et x le vecteur propre associé. Alors, $Ax = \lambda x$ et

$$x^T A x = x^T \lambda x = \lambda (x^T x) \Leftrightarrow \lambda (x^T x) > 0 \Leftrightarrow \lambda > 0.$$

2. Supposons que toutes les valeurs propres λ_i de A sont positives. Comme A est symétrique, il existe une matrice orthogonale Q telle que

$$A = Q \Lambda Q^T,$$

où Λ est la matrice diagonale des valeurs propres. Soit un vecteur $x \neq 0$ de dimension appropriée et $y = Q^T x \neq= 0$. Alors,

$$x^T A x = x^T Q \Lambda Q^T x = y^T \Lambda y = \sum_{i=1}^{d} \lambda_i y_i^2 > 0.$$

Ainsi A est bien définie positive.

□

Comme le déterminant est le produit des valeurs propres, le déterminant d'une matrice définie positive est strictement positif. Une matrice définie positive A est donc nécessairement inversible et son inverse sera aussi définie positive, car alors pour tout $x \neq 0$, $\exists y : Ay = x$ et donc

$$x^T A^{-1} x = (y^T A^T) A^{-1} (Ay) = y^T A y > 0.$$

Une matrice *semi-définie positive* se définit de la même façon, mais avec une inégalité large plutôt que stricte : on demandera que pour tout $x \neq 0$, $x^T Ax \geq 0$. De même, une matrice symétrique sera semi-définie positive si et seulement si toutes ses valeurs propres sont positives ou nulles. Une matrice définie positive est bien sûr également semi-définie positive. On note habituellement par S_d^+ l'ensemble des matrices semi-définies positives de dimension d.

Proposition A.1.2. *Pour tout matrice réelle B, $(BB^T) \in S_d^+$.*

Démonstration. Tous d'abord, notons que $(BB^T)^T = BB^T$, donc la matrice est bien symétrique. Ensuite, pour tout $x \neq 0$, définissons le vecteur $y = B^T x$. Alors,

$$x^T(BB^T)x = (x^T B)(B^T x) = y^T y = \sum_{i=1}^{d} y_i^2 \geq 0.$$

□

Proposition A.1.3 (Linéarité)**.** *Soit A et $B \in S_d^+$ et a,$b > 0$. Alors, $(aA + bB) \in S_d^+$, et elle sera même définie positive si A ou B l'est.*

Démonstration. Soit $x \neq 0$. Alors, $x^T Ax \geq 0$ et $x^T Bx \geq 0$ et donc

$$x^T(aA + bB)x = ax^T Ax + bx^T Bx \geq 0.$$

Si $x^T Ax > 0$ ou $x^T Bx > 0$, on aura alors l'inégalité stricte. □

Ce résultat s'étend bien sûr par induction à n'importe quelle combinaison linéaire à coefficients positifs. Nous pouvons d'ores et déjà vérifier certaines des bonnes propriétés des matrices adaptatives C_t.

Proposition A.1.4. *Considérons l'ensemble des matrices C_t telles que définies en (2.3.1).*

1. *Toute matrice de covariance empirique est semi-définie positive.*
2. *Les matrices C_t sont définies positives.*
3. *Il existe $c_1 > 0$ tel que $C_t - c_1 I \in S_d^+$ pour toute matrice C_t.*

Démonstration. .

1. Soit C la matrice de covariance empirique d'un vecteur $(x_1, \ldots, x_k)$ d'éléments de $\mathbb{R}^d$. Alors par définition,

$$C = \frac{1}{k} \sum_{i=0}^{k} (x_i - \overline{x}_k)(x_i - \overline{x}_k)^T.$$

   Par la proposition A.1.2, C est donc une somme finie de matrices de S_d^+ et est donc dans S_d^+ par la proposition A.1.3.

2. Nous avons $C_t = s_d(C + \epsilon I)$, pour une certaine matrice $C \in S_d^+$, avec $s_d, \epsilon > 0$. Alors, ϵI est bien sûr définie positive et donc C_t l'est aussi par la proposition A.1.3.

3. En posant $c_1 = s_d\epsilon$, nous avons que $C_t - c_1 I = s_d C \in S_d^+$, par ce qui précède.

□

A.1.3. Relation d'ordre et compacité

Tel que vu dans le chapitre 4, il est possible de définir une relation d'ordre partiel sur S_d^+. Pour $A, B \in S_d^+$, on notera $A \leq B$ si $B - A \in S_d^+$. On appelle cette relation l'ordre partiel de Löwner. Voyons quelques-unes de ses propriétés.

Proposition A.1.5. *Soient $A, B \in S_d^+$ telles que $A \leq B$, et $\lambda_i(A), \lambda_i(B),\ i \in \{1, \ldots, d\}$, leurs valeurs propres en ordre croissant. Alors,*

1. *Pour tout i, $\lambda_i(A) \leq \lambda_i(B)$;*
2. $\det(A) \leq \det(B)$ *;*
3. $\lambda_1(A)I \leq A \leq \lambda_n(A)I$.

Démonstration. .

1. $A + (B - A) = B$ et $B - A$ est semi-définie positive. Alors, par le théorème de Weyl, pour tout i :
$$\lambda_i(B) \geq \lambda_i(A) + \lambda_i(B - A) \geq \lambda_i(A),$$
puisque $\lambda_i(B - A) \geq 0$ pour tout i.
2. Direct, par le premier volet et puisque le déterminant est le produit des valeurs propres, ici toutes positives ou nulles.
3. Par le théorème de Rayleigh, pour tout $x \neq 0$,
$$\begin{aligned} \frac{x^T A x}{x^T x} \geq \lambda_1(A) &\Leftrightarrow \frac{x^T A x - \lambda_1(A) x^T x}{x^T x} \geq 0 \\ &\Leftrightarrow x^T A x - \lambda_1(A) x^T x \geq 0 \\ &\Leftrightarrow x^T (A - \lambda_1(A) I) x \geq 0. \end{aligned}$$
Ainsi, $(A - \lambda_1(A)I) \in S_d^+$, tel que souhaité. La preuve pour la seconde inégalité est analogue.

□

Grâce à ces propriétés, nous pouvons maintenant valider la seconde inégalité de (4.2.9).

Corollaire A.1.1. *Il existe $c_2 > 0$ tel que pour toute matrice adaptative C_t, $C_t \leq c_2 I$.*

Démonstration. Tel que déjà argumenté, la compacité de l'espace S et la construction des matrices C_t garantit l'existence d'une certaine borne M_S sur tous les coefficients possibles

pour ces matrices. Ceci se traduit par l'existence d'une borne λ^* sur toutes les valeurs propres possibles, par exemple par la relation

$$d^2 M_S^2 \geq \sum_{i,j} (C_t)_{ij}^2 = \mathrm{Tr}(C_t C_t^T) = \mathrm{Tr}(Q_t \Lambda_t Q_t^T Q_t \Lambda_t Q_t^T) = \mathrm{Tr}(\Lambda^2) = \sum_i \lambda_i(C_t)^2,$$

où $Q_t \Lambda_t Q_t^T$ est la diagonalisation de C_t. Alors, en prenant $c_2 = \lambda^*$, nous pouvons conclure par le troisième volet de la proposition A.1.5 que $C_t \leq c_2 I$ pour toute matrice C_t. □

Ce résultat, combiné avec la proposition A.1.4 prouve donc l'existence de constantes $c_1, c_2 > 0$ telles que pour toute matrice C_t,

$$c_1 I \leq C_t \leq c_2 I.$$

Ceci nous assure aussi que l'ensemble des matrices C_t réalisables est contenu dans un ensemble compact. En effet, en considérant par exemple l'espace vectoriel des matrices symétriques en d dimensions normé par la norme de Frobenius :

$$\|A\|_F := \sqrt{\mathrm{Tr}(A^T A)} = \sqrt{\sum_i \lambda_i(A)^2},$$

l'ensemble $\{A : \sqrt{d} c_1 \leq \|A\|_F \leq \sqrt{d} c_2\}$ est compact et contient toutes les réalisations possibles de C_t.

Par ailleurs, par le second volet de la proposition A.1.5, l'inégalité est valable aussi pour les déterminants, ainsi

$$0 < c_1^d \leq \det(C_t) \leq c_2^d. \tag{A.1.1}$$

A.1.4. Résultats de convergence

Les propositions suivantes nous aideront à démontrer la convergence des différences de densités normales associées à la suite (C_t).

Proposition A.1.6. *Pour toute suite de matrices C_t telles que définies en (2.3.1) avec S compact, nous avons :*

1. $C_{t+1} - C_t \to \mathbf{0}$ *coefficient par coefficient quand* $t \to \infty$;
2. $\det(C_{t+1}) - \det(C_t) \to 0$ *quand* $t \to \infty$;
3. $\det(C_t)/\det(C_{t+1}) \to 1$ *quand* $t \to \infty$;
4. $C_{t+1}^{-1} - C_t^{-1} \to \mathbf{0}$ *coefficient par coefficient quand* $t \to \infty$.

Démonstration. Le premier résultat a été montré en (4.2.13). Le second en découle assez directement, puisque le déterminant d'une matrice est constitué d'une somme finie de produits finis de ses coefficients et par opérations sur les limites. Le troisième découle directement du second et du fait que par (A.1.1) la suite $(\det(C_t))$ est bornée inférieurement. Finalement, chacun des coefficients de C_t^{-1} peut être exprimé à un signe près comme le rapport entre un déterminant d'une sous-matrice de C_t et $\det(C_t)$. Chaque coefficient est donc

constitué d'opérations élémentaires sur des éléments de C_t et on conclut une fois encore par opérations sur les limites. □

Proposition A.1.7. *Soit une matrice D_t définie positive telle que $D_t \to \mathbf{0}$ coefficient par coefficient quand $t \to \infty$. Alors, pour tout vecteur z de dimension appropriée,*

$$z^T D_t z \xrightarrow[t\to\infty]{} 0$$

et cette convergence sera uniforme sur n'importe quel ensemble borné de vecteurs z.

Démonstration. Pour tout z fixé,

$$z^T D_t z = \sum_{i,j} z_i z_j (D_t)_{ij} \to 0.$$

Pour une borne $M_z > 0$ et un ensemble Z tel que $z \in Z \Rightarrow \max_i |z_i| \leq M_z$, alors

$$0 \leq \sup_Z |z^T D_t z| \leq \sum_{i,j} \sup_Z |z_i z_j (D_t)_{ij}| = \sum_{i,j} \sup_Z |z_i z_j| |(D_t)_{ij}| \leq \sum_{i,j} M_z^2 |(D_t)_{ij}| \to 0.$$

□

A.2. Propriétés de la densité normale

Rappelons d'abord la forme de la densité d'une loi normale multivariée en d dimensions de moyenne $\mu \in \mathbb{R}^d$ et de matrice de covariance $\Sigma \in \mathbb{R}^{d\times d}$:

$$f(y|\mu,\Sigma) = \frac{1}{(2\pi)^{d/2}\det(\Sigma)^{1/2}} e^{-\frac{1}{2}(y-\mu)^T\Sigma^{-1}(y-\mu)}.$$

Pour que f soit bien définie et qu'il s'agisse bien d'une densité, Σ doit être définie positive. La densité f est alors continue et strictement positive sur son support $\mathbb{R}^d$. Notons aussi qu'elle ne dépend de y et de μ qu'à travers leur différence.

Dans le cas des algorithmes adaptatifs, la densité à une étape donnée aura comme moyenne une certaine valeur x actuelle de l'échantillon et comme matrice de covariance C_t, nous aurons alors

$$f_{\Gamma_t}(x,y) := f(y|x,C_t) = \frac{1}{(2\pi)^{d/2}\det(C_t)^{1/2}} e^{-\frac{1}{2}(y-x)^T C_t^{-1}(y-x)}.$$

Proposition A.2.1. *Pour un espace échantillonnal S compact et les matrices C_t définies comme en (2.3.1), $f_{\Gamma_t}(x,y)$ est bornée uniformément pour tous $x,y \in S$, $\Gamma_t \in \mathcal{G}$.*

Démonstration. Comme $(y-x)^T C_t^{-1}(y-x) \geq 0$, et par (A.1.1), il existe $c_1 > 0$ tel que

$$\sup_{S\times\mathcal{G}} f_{\Gamma_t}(x,y) \leq \frac{1}{(2\pi)^{d/2}\inf_{\mathcal{G}}\det(C_t)^{1/2}} \leq \frac{c_1^{d/2}}{(2\pi)^{d/2}}.$$

□

Ceci justifie l'existence de la borne M définie en (4.2.10). Nous terminons cette annexe avec la preuve de la convergence uniforme énoncée en (4.2.14), ce qui terminera de valider les résultats théoriques du chapitre 4.

Proposition A.2.2. *Pour un espace échantillonnal S compact et les matrices C_t définies comme en (2.3.1),*

$$\sup_{x,y\in S} |f_{\Gamma_{t+1}}(x,y) - f_{\Gamma_t}(x,y)| \xrightarrow[t\to\infty]{} 0.$$

Démonstration. Nous montrons d'abord que le rapport des densités successives tend uniformément vers 1. En effet,

$$\frac{f_{\Gamma_{t+1}}(x,y)}{f_{\Gamma_t}(x,y)} = \frac{\det(C_t)^{1/2}}{\det(C_{t+1})^{1/2}} e^{-\frac{1}{2}(y-x)^T(C_{t+1}^{-1}-C_t^{-1})(y-x)},$$

avec $\det(C_t)/\det(C_{t+1}) \to 1$ et $C_{t+1}^{-1} - C_t^{-1} \to \mathbf{0}$ quand $t \to \infty$, par la proposition A.1.6. De plus, puisque l'ensemble des coefficients possibles de $(x-y)$ sur S est borné, nous avons, par la proposition A.1.7, que

$$h_t(x,y) := -\frac{1}{2}(y-x)^T(C_{t+1}^{-1} - C_t^{-1})(y-x) \xrightarrow[t\to\infty]{} 0,$$

uniformément pour tous $x,y \in S$. Alors, par les propriétés de l'exponentielle et en notant que $h_t(x,y) \leq 0$ pour tous t,x,y,

$$\begin{aligned}
\lim_{t\to\infty} \sup_S \left| e^{h_t(x,y)} - 1 \right| &= \lim_{t\to\infty} \inf_S \left(e^{h_t(x,y)} - 1 \right) \\
&= \lim_{t\to\infty} \exp\left(\inf_S h_t(x,y) \right) - 1 \\
&= \exp\left(\lim_{t\to\infty} \inf_S h_t(x,y) \right) - 1 = e^0 - 1 = 0.
\end{aligned}$$

Nous avons donc bien

$$\sup_S \left| \frac{f_{\Gamma_{t+1}}(x,y)}{f_{\Gamma_t}(x,y)} - 1 \right| \xrightarrow[t\to\infty]{} 0,$$

et puisque $f_{\Gamma_t}(x,y)$ est uniformément bornée sur S,

$$\sup_S |f_{\Gamma_{t+1}}(x,y) - f_{\Gamma_t}(x,y)| \leq \sup_S |f_{\Gamma_t}(x,y)| \sup_S \left| \frac{f_{\Gamma_{t+1}}(x,y)}{f_{\Gamma_t}(x,y)} - 1 \right| \xrightarrow[t\to\infty]{} 0.$$

□

Annexe B

ALGORITHME DES K-MOYENNES

Nous avons fait référence à l'algorithme des K-moyennes dans le cadre d'une sous-procédure pouvant être ajoutée à notre algorithme MCMC (voir chapitre 3). Nous présentons ici de façon succincte quelques détails sur cette méthode. On pourra se référer au chapitre 14 de [18] pour de plus amples informations et des références additionnelles.

B.1. Présentation

La méthode des K-moyennes est un algorithme de partitionnement d'un ensemble d'observations en un nombre K prédéfini de groupes. La partition souhaitée est celle minimisant la dispersion intra-groupe des observations. Formellement, pour des observations $x_1, \ldots, x_n \in \mathbb{R}^d$, une partition sera représentée par une fonction $C : \mathbb{R}^d \to \{1, \ldots, K\}$ attribuant à chaque observation le numéro de son groupe. La partition C optimale recherchée est donc celle minimisant le critère suivant :

$$W(C) = \sum_{k=1}^{K} \sum_{i<j:C(x_i)=C(x_j)=k} \|x_i - x_j\|_2^2. \tag{B.1.1}$$

De façon équivalente, avec $\overline{x}_k$ la moyenne empiriques des observations du groupe k et N_k le nombre d'observations dans ce même groupe,

$$W(C) = \sum_{k=1}^{K} N_k \sum_{i:C(x_i)=k} \|x_i - \overline{x}_k\|_2^2. \tag{B.1.2}$$

Notons aussi que n'importe quel vecteur $m = (m_1, \ldots, m_K)$ de K éléments distincts de $\mathbb{R}^d$ définit une partition unique C_m de la façon suivante :

$$C_m(x) = \underset{k \in \{1,\ldots,K\}}{\arg\min} \|x - m_k\|_2^2$$

et en attribuant par convention la région de plus petit indice en cas d'égalité. Les éléments m_k constituent donc le centre de chacune des régions. Le principe de l'algorithme des K-moyennes est de faire évoluer ce vecteur m de façon à faire diminuer la valeur du critère $W(C_m)$(B.1.2). Voyons maintenant son fonctionnement détaillé.

B.2. Algorithme

Choisir au hasard des valeurs distinctes $m_1^{(0)}, \ldots, m_K^{(0)} \in \mathbb{R}^d$. Puis, à chaque temps t :

1. Calculer $C_{m^{(t)}}(x_i)$ pour chacune des observations ;
2. Choisir comme nouvelles valeurs $m_k^{(t+1)}$ la moyenne empirique $\overline{x}_k^{(t)}$ des observations du groupe k, tel que défini par la partition $C_{m^{(t)}}$ de l'étape 1.
3. Arrêter lorsque deux itérations consécutives laissent la partition inchangée, donc lorsque
$$C_{m^{(t)}}(x_1), \ldots, C_{m^{(t)}}(x_n) = C_{m^{(t+1)}}(x_1), \ldots, C_{m^{(t+1)}}(x_n).$$

Ainsi, l'algorithme effectue tour à tour un calcul des moyennes empiriques des éléments de chaque groupe, puis une ré-attribution des groupe en fonction de ces nouveaux points centraux.

Proposition B.2.1. *La procédure telle que décrite convergera toujours vers un minimum local de $W(C)$.*

Démonstration. En effet, le nombre de partition différentes est fini et chacune des étapes 1 et 2 fait diminuer (ou maintient égale) la valeur du critère (B.1.2), par l'argumentation qui suit.

Pour l'étape 2, notons que la valeur $y = \overline{x}_k^{(t)}$ minimise la quantité
$$\sum_{i:C_{m^{(t)}}(i)=k} \|x_i - y\|_2^2.$$
Ainsi, nous avons
$$\sum_{i:C_{m^{(t)}}(x_i)=k} \|x_i - \overline{x}_k^{(t)}\|_2^2 \leq \sum_{i:C_{m^{(t)}}(x_i)=k} \|x_i - m_k^{(t)}\|_2^2$$
dans chacune des régions.

Pour l'étape 2, si un point x_i change de région par rapport à l'étape précédente, alors ce changement diminue sa contribution au critère, alors que pour les points inchangés, la contribution au critère reste la même.

□

B.3. Considérations pratiques

En général, dans un contexte de classification, on voudra démarrer l'algorithme plusieurs fois avec des valeurs initiales différentes afin de s'approcher du minimum global. Pour l'usage

que nous faisons de cette procédure, ceci n'est bien sûr pas nécessaire, nous avons seulement besoin que le processus s'arrête éventuellement, ce qui sera toujours le cas par ce qui précède.

Tel que déjà mentionné, trouver l'indice k minimisant $\|x-m_k\|_2^2$ pour un point donné peut se faire en comparant la position de x par rapport aux hyperplans séparant les moyennes. Une fois les hyperplans déterminés, cette technique accélère marginalement la classification de chacun des points. Ainsi, pour de grandes valeurs de n et de petites valeurs de K, cette technique sera préférable, puisqu'il y a alors beaucoup de points à classer et peu d'hyperplans à déterminer.

Annexe C

CODE DE SIMULATION

L'essentiel du code utilisé pour les tests de simulations est présenté dans les pages qui suivent. L'essentiel des algorithmes a été implémenté en langage C pour des raisons d'efficacité et les fonctions de haut niveau ont été écrites en langage R afin de faciliter l'interface d'utilisation. Nous donnons tout d'abord quelques informations sur les choix et stratégies numériques utilisés dans la mise en oeuvre de notre algorithme.

C.1. Détails de l'implémentation

La plupart des opérations effectuées par l'algorithme se résument à des manipulations assez simples sur des vecteurs et des matrices. Nous nous attarderons uniquement sur les quelques étapes plus complexes, soit le calcul du rapport de densités, la génération aléatoire des candidats et l'ajustement des quantités adaptatives lors d'une mauvaise exploration initiale de l'espace.

C.1.1. Rapport de densités

Le calcul d'une densité normale multivariée de moyenne μ et de covariance C engendre la forme quadratique suivante :

$$(x-\mu)^T C^{-1}(x-\mu).$$

Il n'est pas souhaitable en pratique d'inverser complètement la matrice C, on privilégiera plutôt une factorisation de Cholesky de C, qui donne une matrice A triangulaire telle que $AA^T = C$. On peut alors résoudre efficacement le système triangulaire $y = A^{-1}(x-\mu)$, puis on obtient

$$(x-\mu)^T C^{-1}(x-\mu) = (x-\mu)^T (A^{-1})^T A^{-1}(x-\mu) = y^t y.$$

Comme la plupart des matrices intervenant dans les rapports de densité sont réutilisées à maintes reprises, l'implémentation de l'algorithme conserve en mémoire ces matrices factorisées. Bien qu'il soit possible en pratique de factoriser et conserver les factorisation C_t, $C_1^{(t)}$,

et $C_2^{(t)}$ uniquement lorsque vraiment nécessaire (c'est-à-dire lors du calcul d'un seuil d'acceptation avec changement de région), nous avons choisi par simplicité de systématiquement calculer ces factorisations à chaque fois que ces matrices sont modifiées, entre autres parce que celles-ci sont utilisées à chaque itération par les algorithmes RAPTOR et OPRA.

C.1.2. Génération des candidats

On peut générer une observation y d'une distribution normale multivariée de vecteur moyen μ et de covariance C à partir d'un vecteur x d'observations i.i.d. d'une distribution normale (0,1) par la relation suivante :

$$y = L^T x + \mu,$$

où L est telle que $LL^T = C$. Il existe plusieurs choix acceptables pour la décomposition L. Nous avons utilisé pour tous les algorithmes comparés la factorisation de Cholesky avec pivotage. Celle-ci est légèrement plus rapide que les autres options et sa stabilité numérique en principe un peu plus faible n'a occasionné aucune perte de précision par rapport aux autres méthodes dans les tests préliminaires effectués. Ici aussi, nous avons choisi de calculer et stocker en mémoire la factorisation de Cholesky avec pivotage des matrices C_t, $C_1^{(t)}$, et $C_2^{(t)}$ systématiquement après chaque modification de ces matrices.

La génération des observations normales (0,1) s'est faite avec l'implémentation C de la fonction *rnorm* du langage R.

C.1.3. Ajustements en cas de sous-exploration

La mise à jour des paramètres adaptatifs peut causer problème lorsque l'une des régions S_k est peu ou pas explorée lors de la période de pré-adaptation. Il est assez simple de contourner ce problème en ajoutant par exemple à l'algorithme une composante qui prolongerait la période de pré-adaptation en cas d'effectifs insuffisants dans l'une des régions. Toutefois, l'on pourrait souhaiter mesurer les performances de l'algorithme sur des temps de pré-adaptation délibérément courts. Dans tous les cas, on évitera tous genres de problèmes en ajoutant les mesures correctives suivantes :

1. Afin d'éviter une covariance nulle lorsque toutes les observations d'une région donnée sont des répliques du même point, on préservera la perturbation ϵI présentée en (2.3.1) lors du premier calcul de covariance.
2. Dans un même ordre d'idées, si une région n'est pas du tout explorée en pré-adaptation, le premier calcul de matrice de covariance se fera dans la portion récursive de l'adaptation et avec seulement deux valeurs, ce qui encore une fois peut causer une matrice nulle (si ces valeurs sont identiques) ou très mal conditionnée (si ces valeurs ne sont pas suffisamment éloignées l'une de l'autre). On évitera tout problème de ce

genre en préservant la perturbation ϵI lors de la mise à jour récursive si l'effectif dans une région donnée est très faible.

3. Afin d'éviter que certains des poids λ_{ij} deviennent nuls après la pré-adaptation, ce qui est problématique puisqu'il s'agit d'un état absorbant pour ces paramètres, on mettra à jour ces paramètres uniquement si une telle mise à jour ne donne que des paramètres non-nuls.

C.2. Code R

Le code R qui suit a servi à préparer et organiser les expériences de simulations présentées dans les sections précédentes. Essentiellement, les fonctions R utilisées sont des fonctions de délégation qui préparent les paramètres, appellent les fonctions C appropriés pour effectuer les calculs, puis organisent les valeurs obtenues pour faciliter la lecture des résultats. Nous présentons dans un premier temps le code source contenant toutes les fonctions R utilisées, suivi d'un exemple de code utilisant ces fonctions pour exécuter l'une des simulations rapportées dans ce mémoire.

C.2.1. Fonctions intermédiaires

```
library(MASS)
library(mvtnorm)#Normale multivariee
library(coda) #Traitement et diagnostics MCMC
library(ellipse) #dessins d'ellipses pour graphiques
library(magic) # Manipulation de matrices
library(plyr)
library(parallel) #traitement en parallele
options("digits"=5)

####################
#1) Utilitaires ###
####################

#taux d'acceptation de chaines paralleles
#chaines : tableau tri-dimensionnel : iterations x dimension x chaines
tauxAcc=function(chaines)
{
  chaines = aperm(chaines,c(2,3,1))
  rep = 0
  dyn.load("RRp.dll")
  res =  .C("tauxAcc", d = as.integer(dim(chaines)[1]), k = as.integer(dim(chaines))[2], n =
       as.integer(dim(chaines))[3], x = chaines, rep = rep )$rep
  dyn.unload("RRp.dll")
  return(res)
}

#=======================================================#

```

```
#calcul du nombre de changements dans les valeurs consecutives d'un vecteur x
switchCount=function(x)
{
  x1=x[-length(x)]
  x2=x[-1]
  sum(x1!=x2)
}

#=========================================================#

#Calcul par simulation des seuils de densite correspondant a des regions de confiance de
    niveau donne, pour une distribution multinormale. La fonction retourne un vecteur de
    seuils de densite, un pour chaque niveau souhaite
#mu1 : moyenne de la premiere composante
#mu2 : moyenne de la seconde composante
#sig1 : covariance de la premiere composante
#sig2 : covariance de la seconde composante
#p : poids de la premiere composante
# fact : facteur de torsion
# seuils : vecteur des niveaux de regions souhaites
#count : nombre de donnees simulees
seuilsMN = function(mu1,mu2,sig1,sig2,p,fact, seuils = c(.5,.9,.95,.99),count = 1000000)
{
  n1 = rbinom(1,count, p)
  x = rbind( rmvnorm(n1,mu1,sig1),rmvnorm(count - n1, mu2, sig2))
  dens = p*dmvnorm(x, mu1,sig1) + (1-p)*dmvnorm(x,mu2,sig2)
  dens = sort(dens, decreasing = T)
  return(c(dens[floor(seuils*count) ]))
}

#=====================================================================#

#generation d'un echantillon de melange de deux normales multivariees
#mu1 : moyenne de la premiere composante
#mu2 : moyenne de la seconde composante
#sig1 : covariance de la premiere composante
#sig2 : covariance de la seconde composante
#p : poids de la premiere composante
# fact : facteur de torsion
#count : nombre de donnees simulees
genMN = function(mu1,mu2,sig1,sig2,p = 1,fact = 0,count = 1000000)
{
  if(p == 1)
  {
    temp = mvrnorm(count, mu1, sig1)
    if(fact == 0)  return(temp)
    temp[,2] = temp[,2] - fact*(temp[,1]^2- 100)
    return(temp)
  }

  n1 = rbinom(1,count, p)
  temp = rbind( rmvnorm(n1,mu1,sig1),rmvnorm(count - n1, mu2, sig2))

```

```
  if(fact ==0) return(temp)
  temp[,2] = temp[,2] - fact*(temp[,1]^2- 100)
  return(temp)
}

#=========================================================#
#Norme au carre d'un vecteur x
jump=function(x) sum(x^2)

###################
#2)Fonctions MCMC#
###################

#mcmc adapatif avec proposition normale et potentiellement 2 regions, potentiellement adaptatives
#type : algo adaptatif choisi : "mh" (aucune adaptation), "am" , "rapt", "rr0" ( opra, pas de recalcul), "rr1" (recalcul apres la pread),
#"rr" (recalcul a toutes les kMoy iterations, ou autres options, voir kMoy), "raptor"
#n : nombre total de points
#t0 : temps de preadaptation
#k : nombre de chaines paralleles
#init : matrice initiale (ou vecteur si k ==1). Une colonne par chaine.
#cible : entier donnant le type de cible: 0 : normale, 1: normale etiree,
#   2: melange de 2 normales, 3 : melange de 2 normales incurvees
#paramCible : liste de parametres de la fonction cible : dans l'ordre la (les moyennes) la (les) covariance(s), le poids, le facteur d'etirement. Lister uniquement les parametres utiles (ex. pour cible 1, la fonction s'attend a : mu1, sigma1, facteur d'etirement)
#C1, C2 : covariances adaptatives initiales pour chaque region
#C : covariance globale, aussi la seule covariance utilisee par les types "mh" et "am"
#beta : poids de la densite globale dans les algorithmes regionnaux
#a,b : parametres du plan separateur initial  a^T x  = b.
#kMoy : indicatrice de calcul iteratif des moyennes a t0. 0 pour aucun recalcul. 1 pour un recalcul a t0
#   si kMoy = -1 : version avec plan pondere par les covariances
#si kMoy < -1 : |kMoy| est le nombre d'adaptations du plan apres t0,
#tout autre entier p : recalcul aux p iterations avec critere d'arret.
#dia : sortie de criteres diagnostics
#seuils : seuils de densite (ou autres selon la cible ) pour le calcul des regions de confiance approximatives

mcmc_reg = function(type,n,t0,k,init,cible,paramCible,C,C1,C2,beta, a,b,kMoy,dia, seuils, muC1, muC2 )
{
if(type == "mh")
  return(mcmc_ad(FALSE,n,t0,k,init, cible, paramCible, C,dia, seuils))
if(type == "am")
  return(mcmc_ad(TRUE,n,t0,k,init, cible, paramCible, C,dia, seuils))
if(type == "rapt")
  return(mcmc_adReg(rr = FALSE,n,t0,k,init,cible,paramCible, C, C1, C2, beta, a, b, kMoy, dia, seuils, muC1, muC2))
if(type == "rr0")
  return(mcmc_adReg(rr = TRUE,n,t0,k,init,cible,paramCible, C, C1, C2, beta, a, b, kMoy = 0, dia, seuils, muC1, muC2))
```

```
if(type == "rr1")
  return(mcmc_adReg(rr = TRUE,n,t0,k,init,cible,paramCible, C, C1, C2, beta, a, b, kMoy = 1,
       dia, seuils, muC1, muC2))
if(type == "rr")
  return(mcmc_adReg(rr = TRUE,n,t0,k,init,cible,paramCible, C, C1, C2, beta, a, b, kMoy, dia
      , seuils, muC1, muC2))
if(type == "raptor")
  return(mcmc_adReg(rr = 2,n, t0,k,init,cible,paramCible, C, C1, C2, beta, a, b, kMoy, dia,
      seuils, muC1, muC2, expo = -1.1))
if(type == "raptorX")
  return(mcmc_adReg(rr = 2,n, t0,k,init,cible,paramCible, C, C1, C2, beta, a, b, kMoy, dia,
      seuils, muC1, muC2, expo = -0.1))
}

#=======================================================#

#algorithmes rapt, opra (anciennement rr) et raptor

# rr : selection du type d'algorithme, 0 pour rapt, 1 pour opra, 2 pour raptor
#autres parametres : voir fonction precedente
#expo : perime

mcmc_adReg = function(rr = 1,n,t0,k,init, cible, paramCible, C,C1,C2, beta, a,b,kMoy, dia,
    seuils, muC1, muC2, expo )
{
  #calcul dimensions et tests
  if(k==1) d = length(init)
  else d = dim(init)[1]
  if(!is.matrix(C)) stop("C not a matrix")
  if(dim(C)[1] !=  dim(C)[2] ) stop("C is not square")
  if(dim(C)[1] != d) stop("dimension of C does not match the initial chain value")
  #ajustement au nb de chaines
  t0 = round(t0/k) + 1
  n = round(n/k)
  ############
  #conteneurs
  x = array(0,c(d,k,n+1))
  cumJump = rep(0,n - t0 + 1)
  if(k==1) x[,,1] = init
  else x[,,1] = init
  #diagnostics : emplacements etc
  region=matrix(as.integer(2),k,n+1)
  regionsR = matrix(as.integer(2),k,n+1)
  regionCand = matrix(as.integer(2),k,n)
  #choix de proposition(1,2 ou 0 si proposition globale)
  prop=matrix(as.integer(3),k,n)
  #poids adaptatifs initiaux
  lambda1 = 0.5
  lambda2 = lambda1
  l1 = 0
  l2 = 0
  moy1 = rep(0,d)
  moy2 = rep(0,d)
```

```
171
172   #parametres cible
173   if(cible < 2)
174   {
175     mu1 = paramCible[[1]]
176     mu2 = mu1
177     sig1 = paramCible[[2]]
178     sig2 = sig1
179     p = 0
180     if(cible == 1) fact = paramCible[[3]]
181     else fact = 0
182   }
183   else
184   {
185     mu1 =   paramCible[[1]]
186     mu2 = paramCible[[2]]
187     sig1 = paramCible[[3]]
188     sig2 = paramCible[[4]]
189     p = paramCible[[5]]
190     if (cible == 3) fact = paramCible[[6]]
191     else fact = 0
192   }
193
194   #generation, appel de fonctions c
195   dyn.load("RRp.dll")
196   if(rr < 2)
197   {
198   out = .C("mcmc_reg", d = as.integer(d), rr = as.integer(rr),n= as.integer(n),t0 = as.
        integer(t0), k =  as.integer(k),cible = as.integer(cible),mu1 = mu1, mu2 = mu2,sig1 =
        sig1,sig2 = sig2,p = p,fact = fact, C = C,C1 = C1, C2 = C2, beta = beta, a = a, b = b,
         kMoy = as.integer(kMoy),dia =  as.integer(dia), x = x, cumJump = cumJump, region =
        region, regionCand = regionCand, prop =  prop,lambda1 = lambda1,lambda2 = lambda2, l1
        = as.integer(l1), l2 = as.integer(l2),moy1 = moy1,moy2 = moy2, regionsR = regionsR)
199   prop = out$prop
200   }
201   else
202   {
203     out = .C("mcmc_raptor", d = as.integer(d), n= as.integer(n),t0 = as.integer(t0), k =  as
          .integer(k),cible = as.integer(cible),mu1 = mu1, mu2 = mu2,sig1 = sig1,sig2 = sig2,p
           = p,fact = fact, C = C, muC1 = muC1, muC2 = muC2, C1 = C1, C2 = C2, beta = beta,dia
           =  as.integer(dia), x = x, cumJump = cumJump, region = region, regionCand =
          regionCand,  lambda = c(lambda1,lambda2), l1 = as.integer(l1), l2 = as.integer(l2),
          moy1 = moy1,moy2 = moy2, regionsR = regionsR, expo = expo)
204     lambda1 = out$lambda[1]
205     lambda2 = out$lambda[2]
206   }
207   x = out$x
208
209 #diagnostics supplementaires
210 if(dia)
211 {
212   tauxCouvCum = matrix(0,4,n+1)
```

```
  tauxCouvCum =  .C("tauxC", d = as.integer(d), k = as.integer(k), n = as.integer(n+1),
      cible = as.integer(cible), mu1 = mu1, mu2 = mu2, sig1 = sig1, sig2 = sig2, p = p, fact
       = fact, x, tauxCouvCum = tauxCouvCum, seuils = seuils, as.integer(4) )$tauxCouvCum
 }
  dyn.unload("RRp.dll")
#sortie
  if(dia) {return(list(Chaines = aperm(x,c(3,1,2)),
                       Matrices = list(C = out$C, C1 = out$C1, C2 = out$C2),
                       Effectifs = c(out$l1, out$l2),
                       Plan = list(a = out$a,b = out$b),
                       Moyennes = rbind(out$moy1, out$moy2),
                       Lambdas = c(out$lambda1, out$lambda2),
                       DistancesC = out$cumJump,
                       DistanceM = out$cumJump[length(out$cumJump)],
                       Regions = t(out$region),
                       RegionsR = t(out$regionsR),
                       RegionsCand = t(out$regionCand),
                       Propositions = t(prop),
                       TauxCouv = t(tauxCouvCum),
                       TauxCouvM = t(tauxCouvCum[,n+1]) ))}
  else return(list(Chaines = x,C = C))
}

##########################################
#3) Sommaires, traitement par lot, etc.##
##########################################

#Calcul et organisation des parametres de performance et d'ajustement de chaines MCMC. La
    fonction lance elle-meme l'algorithme pour pouvoir le chronometrer
#Parametres : voir section 2

sommaireMCMC=function(type,n,t0,k,init,cible,paramCible,C,C1,C2,beta, a,b,kMoy, dia = T,
    seuils, muC1, muC2)
{
    #generation
    t1 = proc.time()
    res=mcmc_reg(type,n,t0,k,init,cible,paramCible,C,C1,C2,beta, a,b,kMoy,dia = T, seuils,
        muC1, muC2)
    t2 = proc.time()
    temps = t2 - t1
    #calculs supplementaires
  if( type == "am") { res$Regions = matrix(2, round(n/k)+ 1 , k) }
  t=table(res$Regions)/length(res$Regions)
  to=table(res$RegionsR)/length(res$RegionsR)
  a =tauxAcc(res$Chaines)

if(cible <= 1)
{
  DistQuadM = jump(apply(res[[1]],2,mean) - (paramCible[[1]]))
}
else
{
```

```
  DistQuadM = jump(apply(res[[1]],2,mean) - (paramCible[[1]]*paramCible[[5]] + paramCible
      [[2]]*(1- paramCible[[5]])))
}

return(c(Temps = temps[[3]],
         DistQuadM = DistQuadM,
         Concordance=sum(res$Regions==res$RegionsR)/n,
         TauxCh_vrai=sum(apply(res$RegionsR,2,switchCount))/n,
         Diff_prop_vrai=abs(to[1]-to[2]),
         Accept=a,
         Jump = res$DistanceM,
         TC50 = res$TauxCouvM[1] - 0.5,
         TC90 = res$TauxCouvM[2] - 0.9,
         TC95 = res$TauxCouvM[3] - 0.95,
         TC99 = res$TauxCouvM[4] - 0.99,
         TauxCh=sum(apply(res$Regions,2,switchCount))/n,
         Diff_prop=abs(t[1]-t[2]),
         TauxPropAutreReg = sum(res$Regions[-dim(res$Regions)[[1]],] != res$RegionsCand)/(n-
             k),
         Norme_C_Globale=norm(res$Matrices[[1]])))
}

#=========================================================#

#calcul de moyennes et ecart-types de statistiques de performance et stockage dans un
    fichier texte
#type : algorithme choisi
#N : nombre de chaines generees pour le calcul
#autres parametres : specification de la cible et des parametres initiaux, voir section 2.
#filePrefix : prefixe du fichier texte de resultats

moyennes = function(type,N,n,t0,k,init,cible,paramCible,C,C1,C2,beta, a,b,kMoy, dia = T,
    seuils, muC1, muC2, filePrefix)
{
  res = replicate(N,sommaireMCMC(type,n,t0,k,init,cible,paramCible,C,C1,C2,beta, a,b,kMoy,
      dia = T, seuils, muC1, muC2))
  resM = cbind(moyenne = apply(res,1,mean,na.rm=T), et = apply(res,1,sd,na.rm=T))
  if(type == "rr") {write.table(resM,paste(filePrefix,type,kMoy,k,t0,n,d,".txt"))    }
  else {write.table(resM,paste(filePrefix,type,k,t0,n,d,".txt"))}
}

#=========================================================#


#traitement par lot : distribution sur plusieurs processeurs de calculs iteratifs de
    differents algorithmes MCMC.
#N : nombre de chaines completes a generer pour chacun des choix d'algorithme
#types : vecteur de choix d'algorithmes, le nombre ne doit pas depasser le nombre de
    processeurs disponibles pour la tache
#autres parametres : specification de la cible et des parametres initiaux, voir section 2.
#filePrefix : prefixe du fichier texte de resultats

```

```
batchSim = function(N,types,n,t0, k, init, cible, paramCible, C, C1, C2, beta, a, b, kMoy,
    dia = T, seuils, muC1, muC2, filePrefix)
{
  cl = makeCluster(length(types), type = "FORK")
  clusterExport(cl,ls(),envir = environment())
  parLapply(cl,as.list(types),moyennes,N=N,n=n, t0 = t0, k=k , init = init, cible = cible,
      paramCible = paramCible,C = C,C1 = C1,C2 = C2,beta = beta,a = a, b = b,kMoy = kMoy,
      dia = dia, seuils = seuils, muC1 = muC1, muC2 = muC2, filePrefix = filePrefix)
  stopCluster(cl)
}

#============================================================#

#Graphique pour algorithme rr en deux dimensions, peut aussi travailler avec raptor.
#type : "rr" ou "raptor"
# parametres suivants : specification de la cible et des parametres initiaux, voir section
    2.
#ech: echantillon d'observations de la distribution cible, obtenus prealablement

graphiques_2df=function(type,n,t0,k,init,cible,paramCible,C,C1,C2,beta, a,b,kMoy, seuils,
    muC1, muC2, ech)
{
  sommaire=sommaireMCMC2(type,n,t0,k,init,cible,paramCible,C,C1,C2,beta, a,b,kMoy, dia = T,
      seuils, muC1, muC2)
  par(mfrow=c(1,1),mar=c(5,5,1,4))
  sd = 2.38^2/length(a)
  res = sommaire[[2]]

  plot(ech, col = rgb(0,0,0,.5),pch=16,cex=.3,xlab="X1",ylab="X2")

  if (type != "raptor")
  {
  if(a[1]==0) abline(h=b/a[2],col="orange")
  if(a[2]==0) abline(v=b/a[1],col="orange")
  else abline(a= b/a[2],b=-a[1]/a[2],col="orange")
  abline(b= -res$Plan$a[1]/res$Plan$a[2], a= res$Plan$b/res$Plan$a[2], col="green")
  }
  lines(ellipse(res$Matrices$C/sd, centre= (res$Effectifs[1]* res$Moyennes[1,] + res$
      Effectifs[2]* res$Moyennes[2,])/(res$Effectifs[1] + res$Effectifs[2]) ),col="blue")
  lines(ellipse(res$Matrices$C1/sd,centre=res$Moyennes[1,]),col="blue")
  lines(ellipse(res$Matrices$C2/sd,centre=res$Moyennes[2,]),col="blue")
  points(res$Moyennes, pch=4, col="green")
  points(t(init),pch=2,col="orange")
  return(sommaire)
}

#============================================================#

#Calcul et organisation des parametres de performance et d'ajustement de chaines MCMC
#Cette fonction retourne egalement l'ensemble de l'echantillon et des mesures generes par la
    fonction mcmc_reg
#La fonction lance elle-meme l'algorithme pour pouvoir le chronometrer
#Parametres : voir fonctions de la section 2
```

```
sommaireMCMC2=function(type,n,t0,k,init,cible,paramCible,C,C1,C2,beta, a,b,kMoy, dia = T,
    seuils, muC1, muC2)
{
    #generation
    t1 = proc.time()
    res=mcmc_reg(type,n,t0,k,init,cible,paramCible,C,C1,C2,beta, a,b,kMoy,dia = T, seuils,
        muC1, muC2)
    t2 = proc.time()
    temps = t2 - t1
  if( type == "am") { res$Regions = matrix(2, round(n/k)+ 1 , k) }
  t=table(res$Regions)/length(res$Regions)
  to=table(res$RegionsR)/length(res$RegionsR)
  a =tauxAcc(res$Chaines)

if(cible <= 1)
{
  DistQuadM = jump(apply(res[[1]],2,mean) - (paramCible[[1]]))
}
else
{
  DistQuadM = jump(apply(res[[1]][-(1:t0),,],2,mean) - (paramCible[[1]]*paramCible[[5]] +
      paramCible[[2]]*(1- paramCible[[5]])))
}

return(list(c(Temps = temps[[3]],
              DistQuadM = DistQuadM,
              Concordance=sum(res$Regions==res$RegionsR)/n,
              TauxCh_vrai=sum(apply(res$RegionsR,2,switchCount))/n,
              Diff_prop_vrai=abs(to[1]-to[2]),
              Accept=a,
              Jump = res$DistanceM,
              TC50 = res$TauxCouvM[1] - 0.5,
              TC90 = res$TauxCouvM[2] - 0.9,
              TC95 = res$TauxCouvM[3] - 0.95,
              TC99 = res$TauxCouvM[4] - 0.99,
              TauxCh=sum(apply(res$Regions,2,switchCount))/n,
              Diff_prop=abs(t[1]-t[2]),
              TauxPropAutreReg = sum(res$Regions[-dim(res$Regions)[[1]],] != res$RegionsCand
                  )/(n-k),
              Norme_C_Globale=norm(res$Matrices[[1]])), res))
}
```

C.2.2. Exemple de simulation

```
#rm(list=ls(all=TRUE))
source("Source_RR_Final.R")

#parametres generaux
n = 100000
t0 = 10000
d = 20
k = 4

#cible
cible = 2
paramCible = list( rep(2.5/d,d), -rep(2.5/d, d), diag(1,d), diag(4,d), p = 0.4)

#parametres initiaux
#partages
init = matrix(c(-1,rep(0,d-1),-1,rep(0,d-1),1, rep(0,d-1),1, rep(0,d-1)),d,4)
dia = TRUE
C = diag(100/d,d)
C1 = diag(.1,d)
C2 = diag(.1,d)
#plan initial
a = c(1,rep(0,d-1))
b = 0
kMoy = -1 #pour opra pondere
#raptor
muC1 = c(1,rep(0,d-1))
muC2 = c(-1,rep(0,d-1))

#seuils de densite pour les regions de confiance
seuils = seuilsMN(rep(2.5/d,d), -rep(2.5/d, d), diag(1,d), diag(4,d), p = 0.4,fact=0, seuils
     = c(.5,.9,.95,.99),count = 1000000)
options(scipen = 0)

#simulations
N = 10000
filePrefix = "normaleUniVI"
#Avec 4 processeurs paralleles
batchSim(N,c("rapt", "rr0","raptorX","rr"),n,t0, k, init, cible, paramCible, C, C1, C2, beta
    , a, b, kMoy, dia = T, seuils, muC1, muC2, filePrefix)
#Sinon,
#moyennes("raptorX",N=100,n,t0,k,init,cible,paramCible,C,C1,C2,beta, a,b,kMoy, dia = T,
    seuils, muC1, muC2, filePrefix)



#edition et exportation
results = read.table(paste(filePrefix,"rapt",k,t0,n,d,".txt"),header=T)
results = cbind(results,read.table(paste(filePrefix,"raptorX",k,t0,n,d,".txt"),header=T))
results = cbind(results,read.table(paste(filePrefix,"rr0",k,t0,n,d,".txt"),header=T))
results = cbind(results,read.table(paste(filePrefix,"rr",kMoy,k,t0,n,d,".txt"),header=T))
```

```
48 write.table(results,paste(filePrefix,k,t0,n,d,".txt"))
```

Annexe D

RÉSULTATS NUMÉRIQUES COMPLETS

Les pages suivantes regroupent, par souci de complétude, les résultats de toutes les simulations de grande taille ayant mené à des observations intéressantes. Les éléments rapportés à la section 6.4 sont un sous-ensemble de ce qui est présenté ici. On pourra se référer à la section 6.1 pour un rappel des nombreux paramètres, critères, algorithmes et symboles utilisés.

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
			$d = 5 \quad n = 10^4 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	0.0258	6.6393e-06	0.0362	4.7901e-06	0.0264	9.5570e-06	0.0276	5.3412e-06
EQM_e	0.0111	7.1391e-05	0.0110	7.0785e-05	0.0109	7.0004e-05	0.0109	6.8962e-05
AQV	1.0368	2.4559e-04	1.0578	2.7006e-04	1.0379	2.4537e-04	1.0385	2.5158e-04
Couv. 50% (%)	0.8575	1.9789e-02	1.3166	1.9642e-02	0.8340	1.9550e-02	0.8165	1.9664e-02
Couv. 90% (%)	0.3813	1.1321e-02	0.5805	1.1167e-02	0.3800	1.1262e-02	0.3606	1.1255e-02
Couv. 95% (%)	0.2333	7.8618e-03	0.3404	7.6663e-03	0.2296	7.8399e-03	0.2153	7.8245e-03
Couv. 99% (%)	0.0619	3.2178e-03	0.0880	3.1396e-03	0.0625	3.1834e-03	0.0579	3.2042e-03
T_a	0.3407	9.2734e-05	0.3338	1.0990e-04	0.3411	9.3130e-05	0.3351	1.1777e-04
			$d = 20 \quad n = 10^5 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	2.3071	1.4761e-03	2.9216	8.3671e-04	2.3061	1.1711e-03	2.3712	1.4231e-03
EQM_e	0.0132	4.2565e-05	0.0134	4.3217e-05	0.0131	4.2047e-05	0.0141	4.6106e-05
AQV	1.2623	8.5680e-05	1.2502	8.5772e-05	1.2624	8.5790e-05	1.2418	1.1143e-04
Couv. 50% (%)	2.2313	1.1172e-02	2.0955	1.1041e-02	2.2094	1.1239e-02	1.9759	1.1272e-02
Couv. 90% (%)	0.9256	5.8913e-03	0.7920	5.9417e-03	0.9207	5.9753e-03	0.8333	5.9212e-03
Couv. 95% (%)	0.5396	3.9793e-03	0.4533	4.0193e-03	0.5353	4.0013e-03	0.4871	4.0156e-03
Couv. 99% (%)	0.1377	1.5444e-03	0.1117	1.5425e-03	0.1361	1.5350e-03	0.1277	1.5422e-03
T_a	0.2674	2.6209e-05	0.2834	4.0556e-05	0.2677	2.5975e-05	0.2816	6.9708e-05
			$d = 20 \quad n = 10^5 \quad t_0 = 10^4 \quad N = 10^4$					
Temps (s)	2.1226	1.9644e-03	2.9814	5.4297e-03	2.1092	3.9597e-04	2.4452	5.7275e-03
EQM_e	0.0140	4.4428e-05	0.0170	5.4263e-05	0.0140	4.5046e-05	0.0139	4.4459e-05
AQV	1.2106	8.1721e-05	1.1877	9.7947e-05	1.2106	8.3691e-05	1.2101	8.3557e-05
Couv. 50% (%)	1.0800	1.1367e-02	0.6626	1.2390e-02	1.0690	1.1595e-02	0.9302	1.1637e-02
Couv. 90% (%)	0.4975	6.2146e-03	0.2651	6.6826e-03	0.5020	6.2945e-03	0.4323	6.3756e-03
Couv. 95% (%)	0.2983	4.2145e-03	0.1577	4.5025e-03	0.3016	4.2333e-03	0.2599	4.3036e-03
Couv. 99% (%)	0.0808	1.6499e-03	0.0412	1.7461e-03	0.0816	1.6293e-03	0.0703	1.6636e-03
T_a	0.2659	2.5051e-05	0.2915	4.8143e-05	0.2661	2.5553e-05	0.2641	3.3390e-05
			$d = 50 \quad n = 2 \cdot 10^5 \quad t_0 = 10^4 \quad N = 10^3$					
Temps (s)	32.9151	2.7978e-02	44.7172	6.0209e-02	33.8018	1.3089e-01	33.4221	1.2294e-02
EQM_e	0.0423	2.7973e-04	0.0476	4.8417e-04	0.0424	2.7086e-04	0.0443	2.9995e-04
AQV	1.2795	1.9896e-04	1.2405	3.7615e-03	1.2797	1.8900e-04	1.2769	1.9712e-04
Couv. 50% (%)	6.5054	3.7036e-02	5.5101	6.6740e-02	6.4996	3.7972e-02	5.5313	3.8994e-02
Couv. 90% (%)	2.4718	1.7961e-02	2.0276	2.4707e-02	2.4738	1.7698e-02	2.1804	1.8600e-02
Couv. 95% (%)	1.3921	1.1705e-02	1.1447	1.5350e-02	1.4025	1.1504e-02	1.2397	1.2011e-02
Couv. 99% (%)	0.3328	4.0638e-03	0.2756	5.0512e-03	0.3445	4.1654e-03	0.3054	4.3408e-03
T_a	0.2533	5.6501e-05	0.2782	7.7208e-04	0.2535	5.7450e-05	0.2528	9.6509e-05

TABLEAU D. I. Résultats comparatifs complets, cible normale unimodale sphérique : $\mu_1 = (0, \ldots, 0), \Sigma_1 = I_d$. Paramètres : $k = 4, \mu_1^{(0)} = X_0^{(1)} = X_0^{(2)} = (-5/d, 0, \ldots, 0), \mu_2^{(0)} = X_0^{(3)} = X_0^{(4)} = (5/d, 0, \ldots, 0), C_1^{(0)} = C_2^{(0)} = 0.1 I_d, C_S^{(0)} = 100 I_d/d$, Plan initial : $x_1 = 0$.

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
			$d = 5 \quad n = 10^4 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	0.0337	1.6032e-05	0.0486	1.7488e-05	0.0345	1.3860e-05	0.0364	1.4967e-05
EQM_e	0.7198	8.4064e-03	0.6702	7.5415e-03	0.7299	8.4658e-03	0.6346	7.2945e-03
AQV	12.9881	1.1079e-02	16.0990	1.4929e-02	12.8885	1.0929e-02	15.6640	1.2811e-02
Couv. 50% (%)	1.9723	1.9489e-02	2.2938	1.9824e-02	1.9610	1.9439e-02	1.9385	1.9651e-02
Couv. 90% (%)	0.7723	1.0859e-02	0.9309	1.0858e-02	0.7566	1.0922e-02	0.7463	1.0730e-02
Couv. 95% (%)	0.4463	7.4455e-03	0.5356	7.4378e-03	0.4325	7.5203e-03	0.4283	7.3877e-03
Couv. 99% (%)	0.1133	2.9809e-03	0.1347	2.9913e-03	0.1099	3.0243e-03	0.1077	3.0126e-03
T_a	0.3470	9.1819e-05	0.3549	1.1216e-04	0.3471	9.1531e-05	0.3441	1.0925e-04
			$d = 20 \quad n = 10^5 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	3.2722	1.6030e-03	4.3143	1.9809e-03	3.2943	1.6495e-03	3.3664	1.5531e-03
EQM_e	0.2151	2.6563e-03	0.1998	2.2805e-03	0.2194	2.6524e-03	0.1375	1.6956e-03
AQV	4.5477	1.9506e-03	5.8049	3.5005e-03	4.5092	1.8796e-03	6.1889	3.2964e-03
Couv. 50% (%)	2.3909	1.1122e-02	2.2442	1.1034e-02	2.4059	1.1155e-02	2.1859	1.1290e-02
Couv. 90% (%)	0.9726	5.8732e-03	0.8270	5.9057e-03	0.9745	5.8815e-03	0.8929	5.9361e-03
Couv. 95% (%)	0.5642	3.9493e-03	0.4718	4.0175e-03	0.5651	3.9656e-03	0.5192	3.9794e-03
Couv. 99% (%)	0.1430	1.5211e-03	0.1181	1.5604e-03	0.1425	1.5371e-03	0.1325	1.5210e-03
T_a	0.2681	2.6289e-05	0.2859	4.1240e-05	0.2680	2.6007e-05	0.2732	5.3529e-05
			$d = 20 \quad n = 10^5 \quad t_0 = 10^4 \quad N = 10^4$					
Temps (s)	3.0186	1.3067e-03	4.0994	1.9627e-03	3.0396	1.6406e-03	3.1080	1.2889e-03
EQM_e	0.4385	5.0084e-03	0.4684	4.6928e-03	0.4435	5.0249e-03	0.3716	4.2198e-03
AQV	4.1436	2.1639e-03	5.0741	3.6903e-03	4.1166	2.1347e-03	5.0983	2.9185e-03
Couv. 50% (%)	1.4600	1.1681e-02	1.1098	1.2243e-02	1.4656	1.1760e-02	1.3545	1.1560e-02
Couv. 90% (%)	0.6703	6.2497e-03	0.4518	6.5260e-03	0.6661	6.1555e-03	0.6164	6.1982e-03
Couv. 95% (%)	0.4000	4.1971e-03	0.2645	4.3738e-03	0.3960	4.1562e-03	0.3687	4.1903e-03
Couv. 99% (%)	0.1058	1.6123e-03	0.0691	1.6735e-03	0.1077	1.5999e-03	0.1007	1.6172e-03
T_a	0.2677	2.5499e-05	0.2939	4.8327e-05	0.2677	2.5358e-05	0.2669	3.3445e-05
			$d = 50 \quad n = 2 \cdot 10^5 \quad t_0 = 10^4 \quad N = 10^3$					
Temps (s)	42.8289	5.6423e-02	56.7632	9.9536e-02	42.8397	6.5971e-02	43.4046	5.2014e-02
EQM_e	0.5137	1.8581e-02	0.6840	2.4418e-02	0.5752	1.8383e-02	0.3473	1.2701e-02
AQV	2.3931	3.0999e-03	2.6304	1.5650e-02	2.3871	2.9990e-03	2.8513	4.5295e-03
Couv. 50% (%)	6.8135	3.9836e-02	6.1825	1.0193e-01	6.8060	3.9577e-02	6.0684	3.8504e-02
Couv. 90% (%)	2.6015	1.7822e-02	2.2466	3.4089e-02	2.5821	1.8229e-02	2.3439	1.8784e-02
Couv. 95% (%)	1.4746	1.1516e-02	1.2693	1.9562e-02	1.4678	1.1838e-02	1.3414	1.1992e-02
Couv. 99% (%)	0.3594	4.3492e-03	0.3046	5.5356e-03	0.3554	4.2448e-03	0.3274	4.3504e-03
T_a	0.2546	6.1770e-05	0.2716	1.3502e-03	0.2545	5.9445e-05	0.2549	1.0906e-04

TABLEAU D. II. Résultats comparatifs complets, cible normale unimodale étirée : $\mu_1 = (0, \ldots, 0), \Sigma_1 = \text{diag}(100,1,\ldots,1)$. Paramètres : $k = 4, \mu_1^{(0)} = X_0^{(1)} = X_0^{(2)} = (5{-}5/d,0,\ldots,0), \mu_2^{(0)} = X_0^{(3)} = X_0^{(4)} = (5{+}5/d,0,\ldots,0), C_1^{(0)} = C_2^{(0)} = 0.1 I_d, C_S^{(0)} = 100 I_d/d$, Plan initial : $x_1 = 5$.

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
			$d = 5 \quad n = 10^4 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	0.0258	8.5553e-06	0.0361	8.8491e-06	0.0263	7.5639e-06	0.0276	8.5848e-06
EQM_e	24.2203	9.8202e-02	25.3926	1.1054e-01	24.6139	9.9524e-02	26.7774	1.0499e-01
AQV	3.2843	5.5353e-03	2.6256	3.3064e-03	3.1462	4.9793e-03	2.8048	4.3722e-03
Couv. 50% (%)	5.2674	3.2171e-02	5.3842	4.0040e-02	5.2949	3.2119e-02	5.5483	3.6416e-02
Couv. 90% (%)	1.2312	1.4460e-02	1.3025	1.8570e-02	1.2434	1.4525e-02	1.2960	1.6697e-02
Couv. 95% (%)	0.0441	9.4684e-03	0.1082	1.2157e-02	0.0408	9.5561e-03	0.0623	1.0902e-02
Couv. 99% (%)	-1.0965	4.2912e-03	-1.0212	5.1169e-03	-1.0946	4.2729e-03	-1.0905	4.6770e-03
T_a	0.1830	1.5249e-04	0.1397	1.4221e-04	0.1799	1.6274e-04	0.1552	1.5922e-04
			$d = 20 \quad n = 10^5 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	2.2599	2.8281e-04	2.9164	6.8206e-04	2.2731	2.9410e-04	2.3301	3.0369e-04
EQM_e	19.5825	6.8678e-02	16.3816	7.8152e-02	19.9421	7.0174e-02	23.7867	7.6737e-02
AQV	1.8177	1.2058e-03	1.5213	7.0308e-04	1.7788	1.0985e-03	1.5962	8.7374e-04
Couv. 50% (%)	5.5200	1.5024e-02	4.3793	1.9155e-02	5.5544	1.5431e-02	5.6490	1.7547e-02
Couv. 90% (%)	2.0867	7.2131e-03	1.5786	9.5803e-03	2.0999	7.1928e-03	2.1355	8.2321e-03
Couv. 95% (%)	1.0888	4.7191e-03	0.7611	6.2921e-03	1.0955	4.7268e-03	1.1092	5.3425e-03
Couv. 99% (%)	0.0897	1.7912e-03	-0.0447	2.4351e-03	0.0915	1.8046e-03	0.0909	2.0279e-03
T_a	0.1550	9.7956e-05	0.1149	1.1803e-04	0.1530	1.0514e-04	0.1300	1.2782e-04
			$d = 20 \quad n = 10^5 \quad t_0 = 10^4 \quad N = 10^4$					
Temps (s)	2.0954	2.7315e-04	2.7956	2.9645e-04	2.1071	2.8153e-04	2.2134	1.5130e-03
EQM_e	19.7186	7.2539e-02	18.3285	8.2422e-02	19.9106	7.3231e-02	20.9461	8.4415e-02
AQV	1.7333	1.3068e-03	1.4227	7.0748e-04	1.6993	1.1246e-03	1.5193	9.8754e-04
Couv. 50% (%)	4.2672	1.5375e-02	3.1787	2.0217e-02	4.2674	1.5242e-02	3.8859	1.8009e-02
Couv. 90% (%)	1.6865	7.4368e-03	1.1750	9.8063e-03	1.6926	7.3616e-03	1.5306	8.7713e-03
Couv. 95% (%)	0.8718	4.8692e-03	0.5414	6.4435e-03	0.8749	4.8743e-03	0.7820	5.7198e-03
Couv. 99% (%)	0.0440	1.8372e-03	-0.0984	2.4331e-03	0.0451	1.8537e-03	0.0201	2.1340e-03
T_a	0.1644	1.0305e-04	0.1513	1.1680e-04	0.1624	1.1016e-04	0.1422	1.2916e-04
			$d = 50 \quad n = 2 \cdot 10^5 \quad t_0 = 10^4 \quad N = 10^3$					
Temps (s)	32.8377	2.8269e-02	43.6053	3.4642e-02	32.8733	2.1112e-02	33.2980	7.0866e-03
EQM_e	33.3458	2.2474e-01	27.2430	2.4339e-01	33.3240	2.2415e-01	36.1961	2.3272e-01
AQV	1.3758	1.0702e-03	1.2198	4.7294e-03	1.3674	8.8923e-04	1.3427	1.1385e-03
Couv. 50% (%)	10.2879	4.4969e-02	8.7374	8.2639e-02	10.1773	4.5731e-02	9.4461	4.9672e-02
Couv. 90% (%)	3.6125	1.8525e-02	3.0517	3.0105e-02	3.5817	1.8687e-02	3.3109	2.0693e-02
Couv. 95% (%)	1.9492	1.2037e-02	1.6191	1.8120e-02	1.9251	1.2174e-02	1.7722	1.3585e-02
Couv. 99% (%)	0.3585	4.4101e-03	0.2677	5.8793e-03	0.3493	4.4195e-03	0.3109	5.0854e-03
T_a	0.1935	2.7006e-04	0.1741	5.9588e-04	0.1925	2.6917e-04	0.1788	3.4516e-04

TABLEAU D. III. Résultats comparatifs complets, cible normale unimodale étirée avec torsion : $\mu_1 = (0,\dots,0), \Sigma_1 = \text{diag}(100,1,\dots,1), \psi = 0.1$. Paramètres : $k = 4, X_0^{(1)} = \cdots = X_0^{(4)} = (0,\dots,0), \mu_1^{(0)} = (-0.1,0,\dots,0), \mu_2^{(0)} = (0.1,0,\dots,0), C_1^{(0)} = C_2^{(0)} = 0.1 I_d, C_S^{(0)} = I_d$, Plan initial : $x_1 = 0$.

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
			$d = 5 \quad n = 10^4 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	0.0445	1.0084e-04	0.0657	2.2841e-04	0.0521	8.4280e-05	0.0541	9.9152e-05
EQM_e	30.6938	1.1679e-01	25.3926	1.1054e-01	26.1424	1.0201e-01	28.0096	1.0818e-01
AQV	2.5944	3.5012e-03	2.6256	3.3064e-03	2.9675	4.8824e-03	2.8430	3.8473e-03
Couv. 50% (%)	6.3501	3.7175e-02	5.3842	4.0040e-02	5.6416	3.3140e-02	5.9553	3.5720e-02
Couv. 90% (%)	1.5596	1.6379e-02	1.3025	1.8570e-02	1.3638	1.4614e-02	1.4165	1.5982e-02
Couv. 95% (%)	0.1990	1.0691e-02	0.1082	1.2157e-02	0.1053	9.6456e-03	0.1278	1.0563e-02
Couv. 99% (%)	-1.0622	4.5871e-03	-1.0212	5.1169e-03	-1.0844	4.4334e-03	-1.0806	4.6270e-03
T_a	0.1756	2.4547e-04	0.1397	1.4221e-04	0.1844	1.9180e-04	0.1613	1.7958e-04
			$d = 20 \quad n = 10^5 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	3.2564	7.3813e-04	4.3253	1.3840e-03	3.2903	6.3363e-04	3.3764	7.8599e-04
EQM_e	25.2585	9.0533e-02	16.3816	7.8152e-02	20.4126	7.1024e-02	23.7654	7.8142e-02
AQV	1.5679	6.8154e-04	1.5213	7.0308e-04	1.7449	1.0536e-03	1.5806	7.1156e-04
Couv. 50% (%)	6.1896	1.8460e-02	4.3793	1.9155e-02	5.8180	1.5423e-02	5.8056	1.7389e-02
Couv. 90% (%)	2.3357	8.4254e-03	1.5786	9.5803e-03	2.1797	7.2160e-03	2.1512	8.1978e-03
Couv. 95% (%)	1.2204	5.4322e-03	0.7611	6.2921e-03	1.1380	4.7542e-03	1.1166	5.3808e-03
Couv. 99% (%)	0.1194	2.0128e-03	-0.0447	2.4351e-03	0.0994	1.8130e-03	0.0905	2.0349e-03
T_a	0.1408	1.6397e-04	0.1149	1.1803e-04	0.1560	1.0858e-04	0.1304	1.1311e-04
			$d = 20 \quad n = 10^5 \quad t_0 = 10^4 \quad N = 10^4$					
Temps (s)	3.0128	7.1910e-04	4.1077	1.2151e-03	3.0322	5.9541e-04	3.1334	6.6767e-04
EQM_e	24.9289	9.3628e-02	18.3285	8.2422e-02	20.7963	7.7471e-02	21.9530	8.3982e-02
AQV	1.4983	7.0658e-04	1.4227	7.0748e-04	1.5919	9.2535e-04	1.5218	7.2920e-04
Couv. 50% (%)	4.9693	1.8440e-02	3.1787	2.0217e-02	4.8123	1.6182e-02	4.7098	1.7982e-02
Couv. 90% (%)	1.9526	8.4757e-03	1.1750	9.8063e-03	1.8536	7.5954e-03	1.8140	8.5464e-03
Couv. 95% (%)	1.0186	5.5351e-03	0.5414	6.4435e-03	0.9591	4.9441e-03	0.9378	5.6150e-03
Couv. 99% (%)	0.0798	2.0461e-03	-0.0984	2.4331e-03	0.0611	1.8645e-03	0.0551	2.0862e-03
T_a	0.1479	1.9274e-04	0.1513	1.1680e-04	0.1616	1.6357e-04	0.1415	1.4931e-04
			$d = 50 \quad n = 2 \cdot 10^5 \quad t_0 = 10^4 \quad N = 10^3$					
Temps (s)	42.7070	2.8671e-02	56.8099	6.4008e-02	42.8216	2.7467e-02	43.5063	2.4836e-02
EQM_e	39.5713	2.4068e-01	27.2430	2.4339e-01	33.8121	2.2890e-01	36.2137	2.2480e-01
AQV	1.3316	1.2390e-03	1.2198	4.7294e-03	1.3512	7.9258e-04	1.3448	1.0822e-03
Couv. 50% (%)	11.5616	5.3307e-02	8.7374	8.2639e-02	11.2031	4.7012e-02	10.6810	4.5114e-02
Couv. 90% (%)	3.9098	2.0397e-02	3.0517	3.0105e-02	3.7471	1.9164e-02	3.5902	1.9755e-02
Couv. 95% (%)	2.0874	1.2820e-02	1.6191	1.8120e-02	2.0066	1.2164e-02	1.9170	1.2739e-02
Couv. 99% (%)	0.3796	4.6791e-03	0.2677	5.8793e-03	0.3693	4.3782e-03	0.3476	4.8216e-03
T_a	0.1986	6.0065e-04	0.1741	5.9588e-04	0.1997	2.9903e-04	0.1824	3.4859e-04

TABLEAU D. IV. Résultats comparatifs complets, cible normale unimodale étirée avec torsion et mauvaise partition initiale : $\mu_1 = (0, \ldots, 0), \Sigma_1 = \text{diag}(100,1,\ldots,1), \psi = 0.1$. Paramètres : $k = 4, X_0^{(1)} = \cdots = X_0^{(4)} = (0,\ldots,0), \mu_1^{(0)} = (-0.1,0,\ldots,0), \mu_2^{(0)} = (0.1,0,\ldots,0), C_1^{(0)} = C_2^{(0)} = 0.1 I_d, C_S^{(0)} = I_d$, Plan initial : $x_2 = 0$.

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
			$d = 5 \quad n = 10^4 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	0.0390	1.1066e-04	0.0553	6.5422e-05	0.0401	2.7014e-05	0.0428	3.7857e-05
EQM_e	2.3272	1.6779e-02	2.2799	1.9998e-02	2.0599	1.7636e-02	2.0564	1.6947e-02
AQV	6.2176	5.7386e-03	5.1209	3.9373e-03	4.6684	3.4393e-03	5.2130	3.7175e-03
Couv. 50% (%)	4.4189	3.0494e-02	3.6462	2.9817e-02	3.5803	2.6793e-02	3.6459	2.7929e-02
Couv. 90% (%)	2.2543	1.4801e-02	1.8318	1.5814e-02	1.8733	1.3602e-02	1.8851	1.4424e-02
Couv. 95% (%)	1.3546	9.4971e-03	1.1038	1.0534e-02	1.1467	8.8512e-03	1.1518	9.5183e-03
Couv. 99% (%)	0.3513	3.5103e-03	0.2942	3.9493e-03	0.3106	3.2795e-03	0.3060	3.5188e-03
T_a	0.2322	1.9039e-04	0.2266	2.0036e-04	0.2517	1.6906e-04	0.2324	1.7296e-04
			$d = 20 \quad n = 10^5 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	3.3800	7.3317e-04	4.4242	1.9326e-03	3.3962	7.0679e-04	3.4873	8.4432e-04
EQM_e	1.2553	7.5577e-03	0.9645	7.7396e-03	0.9851	7.3531e-03	1.0634	7.2291e-03
AQV	3.2677	1.2155e-03	2.6945	1.0669e-03	2.3859	8.4177e-04	2.6546	8.4245e-04
Couv. 50% (%)	4.4165	1.4652e-02	2.9336	1.4492e-02	3.8667	1.3100e-02	3.8144	1.3944e-02
Couv. 90% (%)	1.8810	7.2517e-03	1.2535	7.5584e-03	1.6494	6.6367e-03	1.6422	7.1104e-03
Couv. 95% (%)	1.0958	4.7790e-03	0.7344	5.0501e-03	0.9635	4.3897e-03	0.9664	4.6967e-03
Couv. 99% (%)	0.2781	1.7472e-03	0.1878	1.9017e-03	0.2487	1.6417e-03	0.2496	1.7655e-03
T_a	0.1931	9.8872e-05	0.1954	1.0838e-04	0.2083	7.4484e-05	0.1915	8.8142e-05
			$d = 20 \quad n = 10^5 \quad t_0 = 10^4 \quad N = 10^4$					
Temps (s)	3.0404	1.9644e-03	4.1112	2.3451e-03	3.0537	1.9707e-03	3.1497	2.1761e-03
EQM_e	1.3454	9.3948e-03	1.2225	1.0183e-02	1.1482	9.4891e-03	1.2554	9.9476e-03
AQV	3.0526	1.3472e-03	2.5037	1.2190e-03	2.3393	1.2094e-03	2.5585	1.1256e-03
Couv. 50% (%)	2.9929	1.5111e-02	1.8152	1.5628e-02	2.8886	1.4020e-02	2.8977	1.4305e-02
Couv. 90% (%)	1.4157	7.5302e-03	0.8730	8.0200e-03	1.3573	6.9461e-03	1.3610	7.1302e-03
Couv. 95% (%)	0.8471	4.9529e-03	0.5245	5.3087e-03	0.8115	4.5666e-03	0.8115	4.7530e-03
Couv. 99% (%)	0.2252	1.8161e-03	0.1393	1.9669e-03	0.2144	1.7112e-03	0.2147	1.7808e-03
T_a	0.1921	1.0275e-04	0.2232	1.1184e-04	0.2073	1.0007e-04	0.1965	9.9635e-05
			$d = 50 \quad n = 2 \cdot 10^5 \quad t_0 = 10^4 \quad N = 10^3$					
Temps (s)	44.2777	2.7295e-02	56.8833	1.0425e-01	44.4561	3.1596e-02	45.0652	2.7542e-02
EQM_e	2.1909	2.6101e-02	2.1137	4.1433e-02	1.8272	3.1483e-02	2.1315	3.6926e-02
AQV	2.2010	2.7381e-03	1.5037	1.2587e-02	1.7234	1.4714e-03	1.8650	1.8174e-03
Couv. 50% (%)	9.7818	4.4858e-02	8.8190	1.0926e-01	9.0352	4.1220e-02	8.5845	4.2168e-02
Couv. 90% (%)	3.5478	1.8397e-02	3.2784	3.8886e-02	3.3034	1.8247e-02	3.1433	1.8652e-02
Couv. 95% (%)	1.9820	1.1724e-02	1.8567	2.2348e-02	1.8642	1.1469e-02	1.7732	1.1897e-02
Couv. 99% (%)	0.4712	4.1492e-03	0.4514	6.1624e-03	0.4487	4.0829e-03	0.4221	4.2818e-03
T_a	0.2266	2.8329e-04	0.2063	1.4120e-03	0.2311	1.8157e-04	0.2233	1.9109e-04

TABLEAU D. V. Résultats comparatifs complets, cible normale unimodale étirée avec torsion légère et mauvaise partition initiale : $\mu_1 = (0,\ldots,0), \Sigma_1 = \text{diag}(100,1,\ldots,1), \psi = 0.03$. Paramètres : $k = 4, X_0^{(1)} = \cdots = X_0^{(4)} = (0,\ldots,0), \mu_1^{(0)} = (-0.1,0,\ldots,0), \mu_2^{(0)} = (0.1,0,\ldots,0), C_1^{(0)} = C_2^{(0)} = 0.1 I_d, C_S^{(0)} = I_d$, Plan initial : $x_2 = 0$.

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
			$d = 5 \quad n = 10^4 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	0.0465	8.2389e-05	0.0638	1.0450e-04	0.0488	7.8528e-05	0.0506	7.9791e-05
EQM_e	0.0149	1.0683e-04	0.0149	1.0616e-04	0.0154	1.1570e-04	0.0144	1.0273e-04
AQV	1.2534	3.1085e-04	1.2989	3.7882e-04	1.2101	3.1182e-04	1.2678	3.4965e-04
Couv. 50% (%)	0.5222	1.9816e-02	1.0140	1.9723e-02	0.4938	1.9476e-02	0.4581	1.9614e-02
Couv. 90% (%)	0.2894	1.1501e-02	0.5066	1.1150e-02	0.2911	1.1285e-02	0.2494	1.1427e-02
Couv. 95% (%)	0.1891	7.9770e-03	0.3091	7.6614e-03	0.1879	7.8166e-03	0.1654	7.9266e-03
Couv. 99% (%)	0.0549	3.2573e-03	0.0834	3.1334e-03	0.0504	3.2036e-03	0.0481	3.2431e-03
T_a	0.3390	9.2262e-05	0.3261	1.0884e-04	0.3386	9.1661e-05	0.3350	1.2183e-04
T_c	0.7179	1.4324e-04	0.5214	9.6551e-04	0.8846	4.3374e-04	0.6451	1.9940e-04
D_r	0.2209	4.6493e-04	0.2224	4.5864e-04	0.2208	4.9857e-04	0.2220	4.5175e-04
TC_e	0.0671	3.4888e-05	0.0957	6.9771e-05	0.0645	3.6668e-05	0.0137	4.1403e-05
			$d = 20 \quad n = 10^5 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	2.8322	2.5489e-03	3.5165	2.8618e-03	2.8218	2.1755e-03	2.8722	2.2341e-03
EQM_e	0.0133	4.2439e-05	0.0137	4.4637e-05	0.0133	4.2670e-05	0.0144	4.7695e-05
AQV	1.2814	8.6927e-05	1.2703	8.7778e-05	1.2803	8.8058e-05	1.2567	1.2539e-04
Couv. 50% (%)	2.1660	1.1087e-02	2.0555	1.1158e-02	2.1592	1.1077e-02	1.9084	1.1408e-02
Couv. 90% (%)	0.8856	5.8695e-03	0.7519	6.0264e-03	0.8727	5.8760e-03	0.7894	6.0012e-03
Couv. 95% (%)	0.5489	3.9257e-03	0.4635	4.0486e-03	0.5375	3.9898e-03	0.4970	4.0291e-03
Couv. 99% (%)	0.1334	1.5312e-03	0.1091	1.5677e-03	0.1308	1.5598e-03	0.1255	1.5638e-03
T_a	0.2670	2.5824e-05	0.2807	4.0639e-05	0.2673	2.5791e-05	0.2832	7.7728e-05
T_c	0.5785	8.3876e-05	0.5154	1.6079e-03	0.6104	1.1555e-03	0.6630	1.0758e-04
D_r	0.3267	2.0760e-04	0.3262	2.0985e-04	0.3265	2.1317e-04	0.3268	2.1416e-04
TC_e	0.0312	9.2234e-06	0.0003	2.2283e-06	0.0328	9.5105e-06	0.0008	3.6809e-06
			$d = 20 \quad n = 10^5 \quad t_0 = 10^4 \quad N = 10^4$					
Temps (s)	2.6773	2.3826e-03	3.4153	2.8216e-03	2.6489	1.9922e-03	2.7280	2.4052e-03
EQM_e	0.0143	4.5711e-05	0.0180	5.8945e-05	0.0141	4.5348e-05	0.0142	4.5349e-05
AQV	1.2280	8.5275e-05	1.2094	1.0404e-04	1.2276	8.5732e-05	1.2276	8.5768e-05
Couv. 50% (%)	1.0284	1.1414e-02	0.4425	1.2436e-02	1.0278	1.1457e-02	0.8834	1.1607e-02
Couv. 90% (%)	0.4639	6.1957e-03	0.1250	6.7600e-03	0.4654	6.2191e-03	0.3984	6.2686e-03
Couv. 95% (%)	0.3134	4.1895e-03	0.1033	4.5395e-03	0.3151	4.2078e-03	0.2748	4.2271e-03
Couv. 99% (%)	0.0818	1.6270e-03	0.0185	1.7491e-03	0.0810	1.6317e-03	0.0697	1.6483e-03
T_a	0.2660	2.4858e-05	0.2901	4.6193e-05	0.2661	2.4865e-05	0.2642	3.3089e-05
T_c	0.5785	8.6396e-05	0.5189	9.3477e-04	0.6250	4.5049e-04	0.6603	1.2882e-04
D_r	0.3256	2.1704e-04	0.3245	2.4707e-04	0.3252	2.2016e-04	0.3257	2.1533e-04
TC_e	0.0308	8.6697e-06	0.0349	1.3609e-04	0.0320	9.3115e-06	0.0081	8.0497e-06
			$d = 50 \quad n = 2 \cdot 10^5 \quad t_0 = 10^4 \quad N = 10^3$					
Temps (s)	37.5245	7.7803e-02	48.0173	1.1587e-01	37.6422	8.4962e-02	38.0919	8.6536e-02
EQM_e	0.0432	2.7565e-04	0.0480	5.3291e-04	0.0423	2.6817e-04	0.0447	2.8561e-04
AQV	1.2830	1.9047e-04	1.2451	3.8562e-03	1.2830	1.9569e-04	1.2802	2.0102e-04
Couv. 50% (%)	6.5526	3.9905e-02	5.5112	7.1011e-02	6.5586	3.9859e-02	5.5371	3.9773e-02
Couv. 90% (%)	2.4992	1.8090e-02	2.0242	2.5344e-02	2.4909	1.8433e-02	2.1206	1.9060e-02
Couv. 95% (%)	1.4091	1.1865e-02	1.1418	1.5482e-02	1.4055	1.1986e-02	1.1961	1.2427e-02
Couv. 99% (%)	0.3503	4.4125e-03	0.2799	5.0662e-03	0.3509	4.2373e-03	0.3013	4.6677e-03
T_a	0.2532	5.6701e-05	0.2748	7.7164e-04	0.2535	5.6751e-05	0.2526	9.6084e-05
T_c	0.5414	3.3073e-04	0.7030	2.4429e-03	0.5384	1.2657e-03	0.7209	3.4292e-04
D_r	0.4631	6.7268e-04	0.4622	7.1873e-04	0.4635	6.8991e-04	0.4640	6.8546e-04
TC_e	0.0192	1.8224e-05	0.0010	5.0976e-06	0.0211	2.0329e-05	0.0020	1.1785e-05

TABLEAU D. VI. Résultats complets, cible normale bimodale : $\mu_1 = -(2.5,\ldots,2.5)/d, \mu_2 = (2.5,\ldots,2.5)/d, \Sigma_1 = \Sigma_2 = I_d, p = 0.6$. Paramètres : $k = 4, \mu_1^{(0)} = X_0^{(1)} = X_0^{(2)} = (-2{,}0,\ldots,0), \mu_2^{(0)} = X_0^{(3)} = X_0^{(4)} = (2{,}0,\ldots,0), C_1^{(0)} = C_2^{(0)} = 0.1 I_d, C_S^{(0)} = 100 I_d/d$, Plan initial : $x_1 = 0$.

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
			$d = 5 \quad n = 10^4 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	0.0416	1.0278e-04	0.0574	1.1724e-04	0.0422	1.9877e-05	0.0441	2.7440e-05
EQM_e	0.0395	2.6845e-04	0.0420	2.8897e-04	0.0394	2.7893e-04	0.0382	2.6064e-04
AQV	2.3965	1.3879e-03	2.3989	1.5191e-03	2.3719	1.4109e-03	2.3859	1.4345e-03
Couv. 50% (%)	1.7258	3.3187e-02	2.4958	3.3479e-02	1.7628	3.3658e-02	1.6696	3.3321e-02
Couv. 90% (%)	0.8325	1.3883e-02	1.1871	1.3918e-02	0.8384	1.3911e-02	0.8172	1.4097e-02
Couv. 95% (%)	0.4567	8.9595e-03	0.6511	8.9618e-03	0.4551	9.0482e-03	0.4510	9.1368e-03
Couv. 99% (%)	0.1145	3.3611e-03	0.1598	3.2852e-03	0.1121	3.3626e-03	0.1152	3.3154e-03
T_a	0.3047	1.3269e-04	0.3153	1.7967e-04	0.3067	1.3634e-04	0.3125	1.8432e-04
T_c	0.7109	2.1889e-04	0.5047	1.6007e-03	0.7839	4.1187e-04	0.5168	4.0174e-04
D_r	0.1066	6.1088e-04	0.1214	6.3136e-04	0.1076	6.2364e-04	0.1058	6.0612e-04
TC_e	0.0434	3.4782e-05	0.0806	8.5925e-05	0.0491	4.0270e-05	0.0203	5.2466e-05
			$d = 20 \quad n = 10^5 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	3.5274	8.4242e-04	4.6488	1.2748e-03	3.5551	9.2663e-04	3.6259	7.1877e-04
EQM_e	0.0476	1.7047e-04	0.0457	1.5714e-04	0.0468	1.6869e-04	0.0466	1.6738e-04
AQV	2.4235	1.7088e-03	2.3560	1.4991e-03	2.4187	1.6735e-03	2.4438	1.6934e-03
Couv. 50% (%)	8.4275	4.3952e-02	6.9969	4.1908e-02	8.1722	4.3395e-02	7.6128	4.4279e-02
Couv. 90% (%)	2.6274	1.0584e-02	2.1086	1.0595e-02	2.5264	1.0615e-02	2.3948	1.0937e-02
Couv. 95% (%)	1.4387	5.8126e-03	1.1532	5.9495e-03	1.3829	5.8590e-03	1.3156	6.0309e-03
Couv. 99% (%)	0.3340	1.6314e-03	0.2664	1.7597e-03	0.3189	1.6818e-03	0.3068	1.7083e-03
T_a	0.2582	1.0063e-04	0.3032	1.1536e-04	0.2656	1.0404e-04	0.2616	1.1015e-04
T_c	0.6460	2.5920e-04	0.5197	7.7894e-04	0.5682	5.2386e-04	0.5826	4.3911e-04
D_r	0.1772	8.4553e-04	0.1500	7.8216e-04	0.1723	8.2953e-04	0.1622	8.3121e-04
TC_e	0.0209	1.1917e-05	0.0003	2.0756e-06	0.0294	1.6614e-05	0.0038	8.9876e-06
			$d = 20 \quad n = 10^5 \quad t_0 = 10^4 \quad N = 10^4$					
Temps (s)	3.2816	8.7595e-04	4.4251	1.1006e-03	3.3036	6.4243e-04	3.3830	9.6434e-04
EQM_e	0.0469	1.6011e-04	0.0468	1.5853e-04	0.0467	1.5782e-04	0.0462	1.5608e-04
AQV	2.4664	1.8771e-03	2.4503	1.8790e-03	2.4615	1.8928e-03	2.4648	1.8545e-03
Couv. 50% (%)	4.7755	4.8483e-02	3.9636	4.7799e-02	4.8115	4.8908e-02	4.6081	4.8211e-02
Couv. 90% (%)	1.6873	1.2174e-02	1.3545	1.2295e-02	1.6893	1.2334e-02	1.5971	1.2239e-02
Couv. 95% (%)	0.9519	6.7313e-03	0.7600	6.8407e-03	0.9529	6.7765e-03	0.9007	6.7472e-03
Couv. 99% (%)	0.2311	1.9206e-03	0.1856	1.9778e-03	0.2318	1.9158e-03	0.2198	1.9127e-03
T_a	0.2443	1.1417e-04	0.2903	1.5668e-04	0.2467	1.1976e-04	0.2521	1.4303e-04
T_c	0.6269	2.8400e-04	0.5085	4.3139e-04	0.6331	3.2161e-04	0.5491	4.1159e-04
D_r	0.1177	7.7934e-04	0.1067	7.2161e-04	0.1187	7.8493e-04	0.1151	7.6899e-04
TC_e	0.0203	1.2419e-05	0.0101	8.5761e-05	0.0249	1.6789e-05	0.0127	1.6094e-05
			$d = 50 \quad n = 2 \cdot 10^5 \quad t_0 = 10^4 \quad N = 10^3$					
Temps (s)	52.4107	3.8650e-01	62.5131	4.4145e-01	53.9307	4.1217e-01	53.7983	3.9853e-01
EQM_e	0.1476	8.8943e-04	0.2263	1.5408e-03	0.1437	8.7343e-04	0.1457	9.1248e-04
AQV	1.4753	5.8478e-03	0.7249	7.0816e-03	1.5071	6.7895e-03	1.5152	7.0460e-03
Couv. 50% (%)	40.8048	2.3946e-01	46.6756	1.2665e-01	39.4484	2.7311e-01	39.1409	2.7492e-01
Couv. 90% (%)	8.9612	3.1543e-02	9.8757	7.0676e-03	8.7431	3.7089e-02	8.6908	3.8618e-02
Couv. 95% (%)	4.5493	1.4396e-02	4.9574	2.8174e-03	4.4487	1.6978e-02	4.4229	1.7929e-02
Couv. 99% (%)	0.9296	2.6513e-03	0.9954	5.3062e-04	0.9133	3.1448e-03	0.9067	3.4489e-03
T_a	0.2623	4.3011e-04	0.1668	1.4387e-03	0.2660	4.8399e-04	0.2621	5.2826e-04
T_c	0.8054	1.3656e-03	0.6044	7.9911e-03	0.6167	1.2799e-03	0.8440	2.4368e-03
D_r	0.8134	4.7761e-03	0.9322	2.5418e-03	0.7857	5.4451e-03	0.7820	5.4104e-03
TC_e	0.0117	2.7427e-05	0.0010	5.1141e-06	0.0200	3.1313e-05	0.0046	3.2504e-05

TABLEAU D. VII. Résultats complets, cible normale bimodale avec variances inégales : $\mu_1 = -(2.5,\ldots,2.5)/d, \mu_2 = (2.5,\ldots,2.5)/d, \Sigma_1 = 4I_d, \Sigma_2 = I_d, p = 0.6$. Paramètres : $k = 4, \mu_1^{(0)} = X_0^{(1)} = X_0^{(2)} = (-2,0,\ldots,0), \mu_2^{(0)} = X_0^{(3)} = X_0^{(4)} = (2,0,\ldots,0), C_1^{(0)} = C_2^{(0)} = 0.1I_d, C_S^{(0)} = 100I_d/d$, Plan initial : $x_1 = -1$.

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
			$d = 5 \quad n = 10^4 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	0.0664	1.5769e-04	0.0889	2.1601e-04	0.0707	1.5702e-04	0.0767	1.7550e-04
EQM_e	0.0264	1.7125e-04	0.0262	1.6810e-04	0.0259	1.6865e-04	0.0257	1.6176e-04
AQV	1.9459	1.1242e-03	1.9571	1.2007e-03	1.9479	1.1231e-03	1.9594	1.1433e-03
Couv. 50% (%)	1.6264	2.9530e-02	2.1863	2.9863e-02	1.5888	2.9682e-02	1.6014	2.9790e-02
Couv. 90% (%)	0.9482	1.4702e-02	1.2565	1.4373e-02	0.8968	1.4626e-02	0.9211	1.4699e-02
Couv. 95% (%)	0.5420	9.4099e-03	0.7113	9.1937e-03	0.5141	9.4373e-03	0.5288	9.4085e-03
Couv. 99% (%)	0.1359	3.4017e-03	0.1773	3.2432e-03	0.1305	3.4317e-03	0.1365	3.3618e-03
T_a	0.3161	1.5019e-04	0.3172	1.8167e-04	0.3159	1.5199e-04	0.3077	1.7840e-04
T_c	0.5001	1.9716e-04	0.5007	1.4974e-03	0.5008	4.5019e-04	0.6714	2.5609e-04
D_r	0.3441	5.7064e-04	0.3564	5.7142e-04	0.3431	5.6949e-04	0.3435	5.7521e-04
TC_e	0.0606	3.9433e-05	0.0830	8.7305e-05	0.0623	4.0669e-05	0.0163	3.7759e-05
			$d = 20 \quad n = 10^4 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	3.6433	7.9246e-04	4.7757	1.9030e-03	3.6641	1.0643e-03	3.7257	7.6183e-04
EQM_e	0.0302	9.7816e-05	0.0322	1.0459e-04	0.0301	9.7543e-05	0.0317	1.0429e-04
AQV	2.0919	1.3655e-03	2.0526	1.2529e-03	2.0918	1.3643e-03	2.0937	1.4080e-03
Couv. 50% (%)	6.9852	3.7027e-02	6.0620	3.6675e-02	6.9868	3.7032e-02	6.8078	3.8207e-02
Couv. 90% (%)	2.7256	1.1768e-02	2.2652	1.2112e-02	2.7233	1.1717e-02	2.7145	1.2032e-02
Couv. 95% (%)	1.4973	6.4099e-03	1.2422	6.6501e-03	1.4955	6.3778e-03	1.4954	6.5156e-03
Couv. 99% (%)	0.3470	1.7631e-03	0.2897	1.8613e-03	0.3472	1.7684e-03	0.3478	1.7834e-03
T_a	0.2620	9.8054e-05	0.2952	1.1161e-04	0.2618	9.7251e-05	0.2572	1.1182e-04
T_c	0.5001	1.3480e-04	0.5221	1.6507e-03	0.5002	3.3922e-04	0.6773	4.0381e-04
D_r	0.3552	7.8678e-04	0.3337	7.8728e-04	0.3553	7.8641e-04	0.3532	8.1225e-04
TC_e	0.0286	1.1513e-05	0.0003	2.0595e-06	0.0303	1.1843e-05	0.0020	5.4290e-06
			$d = 20 \quad n = 10^5 \quad t_0 = 10^4 \quad N = 10^4$					
Temps (s)	4.7388	1.0458e-02	5.8622	1.4451e-02	4.6639	1.0643e-02	4.7571	1.0971e-02
EQM_e	0.0352	1.1705e-04	0.0369	1.2069e-04	0.0349	1.1611e-04	0.0352	1.1523e-04
AQV	2.1045	1.5227e-03	2.1069	1.5550e-03	2.1043	1.5092e-03	2.1213	1.5628e-03
Couv. 50% (%)	4.2259	4.1512e-02	3.3086	4.1354e-02	4.2438	4.1097e-02	3.9014	4.2044e-02
Couv. 90% (%)	1.9063	1.3496e-02	1.5474	1.3540e-02	1.9033	1.3273e-02	1.7698	1.3470e-02
Couv. 95% (%)	1.0636	7.3813e-03	0.8604	7.4586e-03	1.0587	7.2617e-03	0.9906	7.3483e-03
Couv. 99% (%)	0.2624	2.0006e-03	0.2152	2.0670e-03	0.2635	1.9856e-03	0.2489	2.0014e-03
T_a	0.2488	1.2357e-04	0.2835	1.6209e-04	0.2487	1.2400e-04	0.2425	1.4662e-04
T_c	0.5000	1.3992e-04	0.5176	9.3150e-04	0.5003	2.6755e-04	0.6461	3.9332e-04
D_r	0.2998	8.8141e-04	0.2819	8.7651e-04	0.2998	8.7551e-04	0.2925	8.9124e-04
TC_e	0.0270	1.4210e-05	0.0121	9.6115e-05	0.0287	1.4467e-05	0.0093	1.2556e-05
			$d = 50 \quad n = 2 \cdot 10^5 \quad t_0 = 10^4 \quad N = 10^3$					
Temps (s)	34.3879	3.5111e-02	42.7624	5.4941e-02	34.4780	4.4372e-02	34.9097	3.6428e-02
EQM_e	0.0553	5.1254e-04	0.1081	1.2292e-03	0.0565	5.4570e-04	0.0576	5.4985e-04
AQV	1.4404	5.2328e-03	0.7839	7.6903e-03	1.4476	5.4267e-03	1.4376	5.6263e-03
Couv. 50% (%)	30.4250	1.9586e-01	35.9864	1.3119e-01	30.1101	2.0204e-01	30.0034	2.0183e-01
Couv. 90% (%)	8.7921	3.8640e-02	9.8361	8.8781e-03	8.7585	3.9779e-02	8.8580	4.0274e-02
Couv. 95% (%)	4.4742	1.7541e-02	4.9454	3.4774e-03	4.4632	1.7875e-02	4.5063	1.8202e-02
Couv. 99% (%)	0.9165	3.2711e-03	0.9938	7.0621e-04	0.9174	3.2254e-03	0.9234	3.2454e-03
T_a	0.2615	3.8586e-04	0.1792	1.5699e-03	0.2616	4.0890e-04	0.2562	4.4286e-04
T_c	0.5002	5.0639e-04	0.5434	8.6171e-03	0.5006	9.9433e-04	0.9002	2.1555e-03
D_r	0.8404	4.4720e-03	0.9359	2.4240e-03	0.8363	4.6109e-03	0.8449	4.6196e-03
TC_e	0.0192	2.1400e-05	0.0010	5.0895e-06	0.0210	2.4149e-05	0.0025	1.9568e-05

TABLEAU D. VIII. Résultats complets, cible normale bimodale avec $\mu_1 = \mu_2 = (0, \ldots, 0)$;$\Sigma_1 = I_d, \Sigma_2 = 4I_d, p = 0.6$. Paramètres : $k = 4, \mu_1^{(0)} = X_0^{(1)} = X_0^{(2)} = (-2,0, \ldots, 0), \mu_2^{(0)} = X_0^{(3)} = X_0^{(4)} = (2,0, \ldots, 0), C_1^{(0)} = C_2^{(0)} = 0.1I_d, C_S^{(0)} = 100I_d/d$, Plan initial : $x_1 = 0$.

Algorithme	RAPT		Raptor		OPRA_0		OPRA	
Critère	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type	moyenne	écart-type
			$d = 5 \quad n = 10^4 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	0.0653	1.2501e-04	0.0854	1.7907e-04	0.0724	8.1773e-05	0.0615	1.4221e-04
EQM_e	0.0470	6.0981e-04	0.0446	5.8773e-04	0.0503	6.5560e-04	0.0494	6.3256e-04
AQV	0.4457	8.3927e-04	0.4670	8.8930e-04	0.4374	8.6538e-04	0.4616	8.9853e-04
Couv. 50% (%)	-3.8995	7.6596e-02	-3.2148	8.0366e-02	-3.9455	7.8542e-02	-4.0658	8.1158e-02
Couv. 90% (%)	-0.8695	2.4223e-02	-0.4916	2.3793e-02	-0.8623	2.4720e-02	-0.8978	2.4537e-02
Couv. 95% (%)	-0.3448	1.3978e-02	-0.1483	1.3639e-02	-0.3345	1.4083e-02	-0.3686	1.4062e-02
Couv. 99% (%)	-0.0364	4.3138e-03	-0.0058	4.3024e-03	-0.0307	4.2844e-03	-0.0433	4.3283e-03
T_a	0.2147	1.9478e-04	0.1914	2.0126e-04	0.2212	2.0566e-04	0.1984	2.0077e-04
T_c	0.8327	2.7252e-04	0.5151	2.3784e-03	0.8909	5.6575e-04	0.6251	7.3499e-04
D_r	0.1635	1.1317e-03	0.1779	1.1673e-03	0.1683	1.1579e-03	0.1664	1.1426e-03
TC_e	0.0322	4.5770e-05	0.0451	8.3258e-05	0.0314	6.4268e-05	0.0118	3.9326e-05
			$d = 20 \quad n = 10^5 \quad t_0 = 10^3 \quad N = 10^4$					
Temps (s)	4.1971	9.3595e-03	5.4945	1.1813e-02	4.3097	9.9728e-03	4.4335	1.0314e-02
EQM_e	0.0976	6.5276e-04	0.0889	6.1257e-04	0.0991	6.6136e-04	0.0945	6.6780e-04
AQV	0.2491	1.5184e-03	0.2316	1.1243e-03	0.2527	1.5812e-03	0.2670	1.6896e-03
Couv. 50% (%)	18.0435	1.4054e-01	17.6445	1.2031e-01	18.3338	1.4180e-01	16.7202	1.4983e-01
Couv. 90% (%)	6.1841	3.7034e-02	6.0183	3.2444e-02	6.2623	3.7566e-02	5.7592	4.0134e-02
Couv. 95% (%)	3.2045	1.7991e-02	3.1084	1.5860e-02	3.2445	1.8257e-02	2.9913	1.9571e-02
Couv. 99% (%)	0.6691	3.5976e-03	0.6465	3.2065e-03	0.6768	3.6425e-03	0.6262	3.9157e-03
T_a	0.2246	3.8068e-04	0.2331	4.1139e-04	0.2104	4.0079e-04	0.1969	3.9693e-04
T_c	0.6346	2.3040e-04	0.5074	2.8952e-03	0.7492	2.0536e-03	0.7972	1.7366e-03
D_r	0.6434	2.7847e-03	0.6278	2.5956e-03	0.6489	2.8063e-03	0.6192	2.9210e-03
TC_e	0.0258	4.4824e-05	0.0094	1.1466e-04	0.0185	9.8921e-05	0.0008	6.3768e-06
			$d = 20 \quad n = 10^5 \quad t_0 = 10^4 \quad N = 10^4$					
Temps (s)	3.3849	2.8740e-03	4.5431	3.4228e-03	3.3959	2.8051e-03	3.4669	2.9281e-03
EQM_e	0.0604	5.9282e-04	0.0649	5.7734e-04	0.0611	5.9369e-04	0.0616	5.8570e-04
AQV	0.3786	2.0276e-03	0.3266	1.7702e-03	0.3761	2.0220e-03	0.3837	2.0673e-03
Couv. 50% (%)	5.3423	1.6149e-01	8.2444	1.5547e-01	5.8016	1.6122e-01	5.6138	1.6228e-01
Couv. 90% (%)	2.4845	4.6374e-02	3.3460	4.4243e-02	2.5736	4.6517e-02	2.4824	4.6914e-02
Couv. 95% (%)	1.3761	2.2993e-02	1.7796	2.1964e-02	1.4216	2.3068e-02	1.3672	2.3265e-02
Couv. 99% (%)	0.3069	4.7495e-03	0.3787	4.5743e-03	0.3142	4.7667e-03	0.3011	4.8160e-03
T_a	0.1885	3.9549e-04	0.2190	4.5458e-04	0.1856	3.9498e-04	0.1752	4.0655e-04
T_c	0.6196	2.8266e-04	0.5318	1.3115e-03	0.7450	9.9762e-04	0.6952	1.7175e-03
D_r	0.4160	2.7965e-03	0.4702	2.7867e-03	0.4249	2.7859e-03	0.4210	2.8202e-03
TC_e	0.0203	3.9415e-05	0.0226	1.1879e-04	0.0197	5.7978e-05	0.0079	2.7194e-05

TABLEAU D. IX. Résultats complets, cible normale bimodale avec singularité : $\mu_1 = -(2.5,\ldots,2.5)/d, \mu_2 = (2.5,\ldots,2.5)/d, \Sigma_1 = I_d/10, \Sigma_2 = I_d, p = 0.6$. Paramètres : $k = 4, \mu_1^{(0)} = X_0^{(1)} = X_0^{(2)} = (-2,0,\ldots,0), \mu_2^{(0)} = X_0^{(3)} = X_0^{(4)} = (2,0,\ldots,0), C_1^{(0)} = C_2^{(0)} = 0.1I_d, C_S^{(0)} = 100I_d/d$, Plan initial : $x_1 = 0$.